Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Analýza rádiových spektrogramov metódami nekontrolovaného učenia

Diplomová práca

2022 **Bc. Samuel Jaščur**

Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Analýza rádiových spektrogramov metódami nekontrolovaného učenia

Diplomová práca

Študijný program:	Hospodárska informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra kybernetiky a umelej inteligencie (KKUI)
Školiteľ:	doc. Ing. Peter Butka, PhD.
Konzultant:	RNDr. Šimon Mackovjak, PhD.
	Ing. Viera Maslej Krešňáková, PhD.

Košice 2022 **Bc. Samuel Jaščur**

Abstrakt v SJ

Bleskové výboje generujú elektromagnetické pulzy, ktoré môžu cestovať atmosférou Zeme na veľké vzdialenosti. Namerané pulzy tiež nazývame atmosferiky a ich analýzou je možné skúmať vzdialené búrky ako aj zemskú ionosféru. Použitím metód strojového učenia je možné uľahčiť analýzu nameraných atmosferikov a metódami nekontrolovaného učenia je navyše možné nájsť v nameraných dátach vzory a súvislosti, ktoré by inak mohli ostať nepovšimnuté. Cieľom diplomovej práce je rozšírenie doterajšieho výskumu detekcie nameraných atmosferikov o nové metódy strojového učenia a oboznámenie sa s metódami nekontrolovaného učenia s ich využitím na zhlukovanie dát.

Kľúčové slová

hlboké učenie, neurónová sieť, nekontrolované učenie, zhlukovanie, klasifikácia, atmosferik, tweek, spektrogram

Abstrakt v AJ

Lightning discharges generates electromagnetic pulses that can travel long distances over the Earth's atmosphere. This pulses are also known as atmospherics and by analyzing them it is possible to study distant thunderstorms as well as the Earth's ionosphere. Implementation of machine learning methods can facilitate the analysis of measured atmospherics and by means of unsupervised learning, it is also possible to find patterns and similarities in the measured data that might otherwise go unnoticed. The goal of this diploma thesis is to expand the current research of the detection of measured atmospherics with new methods of machine learning and to get acquainted with the methods of unsupervised learning with their use for data clustering.

Kľúčové slová v AJ

deep learning, neural network, unsupervised learning, clustering, classification, atmospheric, tweek, spectrogram

64231

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra kybernetiky a umelej inteligencie

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**

Študijný program: **Hospodárska informatika**

Názov práce:

Analýza rádiových spektrogramov metódami nekontrolovaného učenia

Analysis of radio spectrograms by unsupervised machine learning techniques

Študent: **Bc. Samuel Jaščur**

Školiteľ: **doc. Ing. Peter Butka, PhD.**

Školiace pracovisko: **Katedra kybernetiky a umelej inteligencie**

Konzultant práce: **RNDr. Šimon Mackovjak, PhD., Ing. Viera Maslej Krešňáková, PhD.**

Pracovisko konzultanta: **Ústav experimentálnej fyziky SAV**

Pokyny na vypracovanie diplomovej práce:

1. Podať teoretický prehľad problematiky využitia metód hlbokého učenia vhodných pre analýzu rádiových spektrogramov, s dôrazom na využitie metód nekontrolovaného učenia pre analýzu rôznych typov bleskov.
2. Analyzovať a predspracovať dostupnú dátovú množinu, extrahovať a automaticky popísať udalosti (blesky) v spektrogramoch.
3. Navrhnúť postupy zhlukovania pre rozlíšenie rôznych typov bleskov v spektrogramoch.
4. Realizovať a vyhodnotiť navrhnuté postupy zhlukovania na zvolenej množine dát.
5. Vypracovať dokumentáciu podľa pokynov katedry a vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 22.04.2022

Dátum zadania diplomovej práce: 29.10.2021

.....

prof. Ing. Liberios Vokorokos, PhD.

dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice 22. 4. 2022 .....

Vlastnoručný podpis

Poďakovanie

Rád by som poďakoval môjmu školiteľovi doc. Ing. Petrovi Butkovi, PhD. za rady s výberom vhodných metód a pripomienky k práci. Ďalej by som chcel poďakovať môjmu konzultantovi RNDr. Šimonovi Mackovjakovi, PhD. za rady a úvod do problematiky v oblasti domény a taktiež veľká vďaka patrí mojej konzultantke Ing. Viere Maslej Krešňákovej, PhD. za veľké množstvo času, skúsenosti a rád, ktoré mi poskytla pri písaní tejto práce. Vďaka patrí aj Ing. Ivane Kolmašovej, PhD. a prof. RNDr. Ondřejovi Santolíkovi, Dr., bez ktorých by táto práca nemohla vzniknúť. Nakoniec by som chcel poďakovať svojej rodine za podporu a vytvorené podmienky, ktoré mi umožnili túto prácu napísať.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov a skratiek

AdaGrad Adaptive Gradient Algorithm

Adam Adaptive Moment Optimalization Algoritm

ANN Artificial Neural Network

BMU Best Matching Unit

CNN Convolutional Neural Network

DL Deep Learning

FN False Negative

FNN Feedforward Neural Network

FP False Positive

ML Machine Learning

NN Neural Network

ReLu Rectified Linear Unit

RMSProp Root Mean Square Propagation

SLP Singlelayer perceptron

SOM Self-organising Map

TN True Negative

TP True Positive

VLF Very Low Frequency

Whistlers Whistler Radio Waves

YOLO You Only Look Once

Úvod

Všetky bleskové výboje generujú elektromagnetické pulzy, ktoré môžu atmosférou Zeme cestovať na veľké vzdialenosti. Analýzou týchto pulzov je možné určiť vzdialenosť od zdroja bleskov a tým skúmať vzdialené búrky ako aj ionosféru (Santolík and Kolmasova, 2017). Takéto analýzy však boli robené iba počas nočných búrok, nikdy nie denných. Autori Santolík and Kolmasova (2017) sú prví, ktorí prezentovali nálezy nezvyčajných atmosferikov so znakmi disperzie počas denných búrok. Využili na to novovzniknutú techniku analýzy pre troj-komponentové elektromagnetické merania. Touto technikou boli schopní určiť zdroj azimutu a taktiež rozdeliť tieto nezvyčajné atmosferiky podľa ich polarity.

Bleskové výboje sa merajú pomocou prístroja ELMAVAN-G. Tento prístroj je klonom prístroja určeného pre družicové merania a bol upravený pre pozemné merania. Namerané výboje sú vo forme spektrogramov, kde každý spektrogram predstavuje meranie dlhé 144 sekúnd. Problémom však je, že takto nameraných dát je obrovské množstvo (približne 700000 spektrogramov za rok) a doposiaľ neexistovala metóda ich automatického spracovania. Práve tu sa nám otvorila možnosť nasadenia techník strojového učenia pre ďalšie spracovanie dát. V našom predchádzajúcom výskume Maslej-Krešňáková et al. (2021) sme pomocou techník strojového učenia dokázali detegovať jednotlivé druhy bleskov automaticky. Metódu sme však ešte nepoužili na väčšie množstvo nameraných dát. To by dokázalo vedcom skúmajúcim tieto javy výrazne uľahčiť prácu a potenciálne dopomôcť k novým objavom. Rozšírením tejto práce by sme navyše mali šancu objaviť skryté vzorce v dátach, ktoré by inak manuálnym prechádzaním mohli ostať nepovšimnuté. Cieľom tejto práce je vytvorenie modelu strojového učenia s využitím konvolučných neurónových sietí, ktorý by dokázal spracovať zachytené atmosferiky a zaklasifikovať ich podľa počtu disperzií. Metódami nekontrolovaného učenia sa následne pokúsime nájsť v dátach vzorce a súvislosti, ktoré by mohli viesť k novým objavom.

Práca je členená do šiestich kapitol. V prvej kapitole sa venujeme teoretickému

rozboru metód strojového učenia. Metódy využité v práci popisujeme detailnejšie a venujeme im viac priestoru. V druhej kapitole sa venujeme doméne a dátam, ktoré máme k dispozícii. Popisujeme v nej samotný merací prístroj, spôsob merania dát, taktiež sa venujeme zloženiu spektrogramov a popisu jednotlivých druhov zachytených udalostí. V tretej kapitole analyzujeme súčasný stav. Zameriavame sa hlavne na výskumy, ktoré využívali metódy, s ktorými pracujeme v našej práci, alebo boli realizované na podobných dátach, ako sú tie naše. V tejto kapitole taktiež popíšeme realizáciu nášho doterajšieho výskumu. V kapitole štyri sa venujeme celému procesu vytvárania modelu klasifikácie disperzií. Popíšeme spôsob predspracovania dát, vytvárania modelu a jeho rôzne obmeny, ktoré sme vytvorili s cieľom zlepšiť presnosť klasifikácie. Jednotlivé obmeny modelu následne porovnáme a vyhodnotíme ich výkonnosť. Ďalej popíšeme proces integrácie vytvoreného modelu s doterajším výskumom a analyzujeme namerané dáta za celý rok. V kapitole päť popíšeme proces aplikácie metód nekontrolovaného učenia na namerané dáta. Vyskúšame metódy zhlukovania k-Means a samo-organizujúcich sa máp, s cieľom nájsť v dátach vzorce a súvislosti. Výsledky metód vhodne interpretujeme a analyzujeme. Zhluky vytvorené metódami porovnáme a popíšeme odlišnosti vyskúšaných metód. V závere práce zhodnotíme dosiahnuté výsledky a načrtneme ďalšie možné smerovanie výskumu.

1 Úvod do strojového učenia

Strojové učenie (*angl. Machine learning*, ML) je rýchlo sa rozvíjajúca podoblasť umelej inteligencie, ktorá sa zaoberá výpočtovými algoritmami. Algoritmy sú navrhnuté napodobňovať ľudskú inteligenciu pomocou učenia z okolitého prostredia. Naučený algoritmus je následne schopný reagovať na vstupy iba na základe naučených informácií a nepotrebuje byť explicitne programovaný. Algoritmy strojového učenia sa delia do troch základných kategórií a to kontrolované učenie (*angl. Supervised learning*), nekontrolované učenie (*angl. Unsupervised learning*) a semi-kontrolované učenie (*angl. Semi-supervised learning*). V práci sa venujeme prvým dvom spomenutým a tie popíšeme podrobnejšie.

Kontrolované učenie funguje na báze učiteľa. Algoritmus sa učí na trénovacej množine dát, ktorá sa skladá z príkladu a triedy, do ktorej daný príklad patrí. Výhodou kontrolovaného učenia je schopnosť produkovať nové výstupy na základe predchádzajúcich skúseností. Správne naučený algoritmus by mal vedieť identifikovať aj príklady z danej triedy, ktoré nikdy nevidel v procese učenia (El Naqa and Murphy, 2015). Dobrým príkladom kontrolovaného učenia sú neurónové siete (*angl. Neural Network*, NN). Ich výhodou je schopnosť extrahovať potrebné príznaky na klasifikáciu zložitých nelineárnych problémov a preto sa dokážu veľmi dobre vysporiadať s akýmkoľvek druhom príkladov. V našej práci používame špeciálny typ NN, ktorý sa nazýva konvolučná neurónová sieť (*angl. Convolutional Neural Network*, CNN). Tento typ neurónovej siete vyniká v spracovaní obrazových dát. Narozdiel od klasickej NN, je si CNN schopná poradiť aj s translačnou invarianciou v obrazových dátach (Ayyadevara, 2018). Nevýhodou kontrolovaného učenia ako celku je potreba anotovanej trénovacej množiny. Vytvoriť takúto množinu je časovo náročné, hlavne ak je potreba anotovať veľký počet tried.

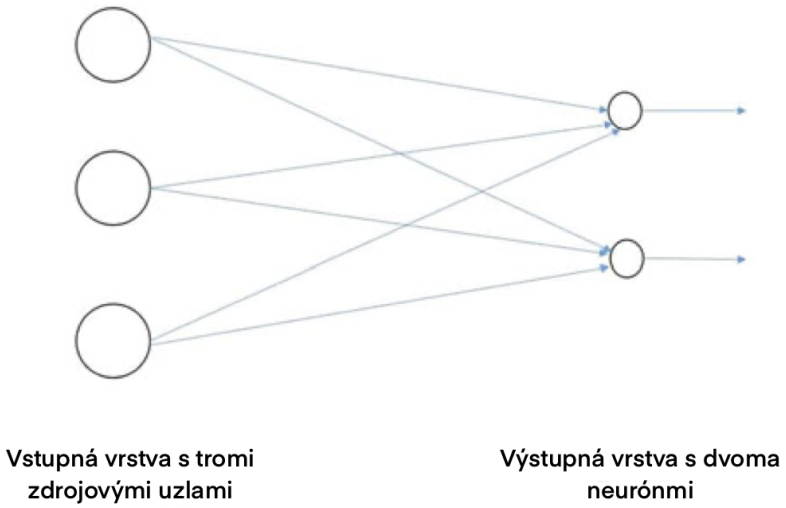
Tento problém eliminujú metódy nekontrolovaného učenia, ktoré nepotrebujú poznať explicitné zaradenie príkladov do tried. Patria sem metódy zhlukovania, ktoré dáta rozdeľujú do zhlukov na základe ich podobnosti. Keďže triedy nie sú pred-

definované, dosiahnuté výsledky môžu ale aj nemusia byť zmysluplné. Môžu však objaviť nové a potenciálne jedinečné vzory v dátach, ktoré by sme kontrolovaným učením nedokázali objaviť. Medzi takéto metódy patria samo-organizujúce sa mapy (*angl. Self Organizing Map*, SOM), k-Means zhlukovanie (*angl. k-Means clustering*) a autoenkódovacie neurónové siete (*angl. Autoencoding Neural Networks*, Autoenkóder). Navyše tieto metódy je možné kombinovať. Napríklad použitie k-Means spolu s autoenkóderom môže pri veľkom objeme dát znížiť početnosť dimenzií pri zachovaní informačnej hodnoty, čo vedie k urýchleniu spracovania dát (Dougherty, 2013).

1.1 Kontrolované učenie pomocou konvolučných neurónových sietí

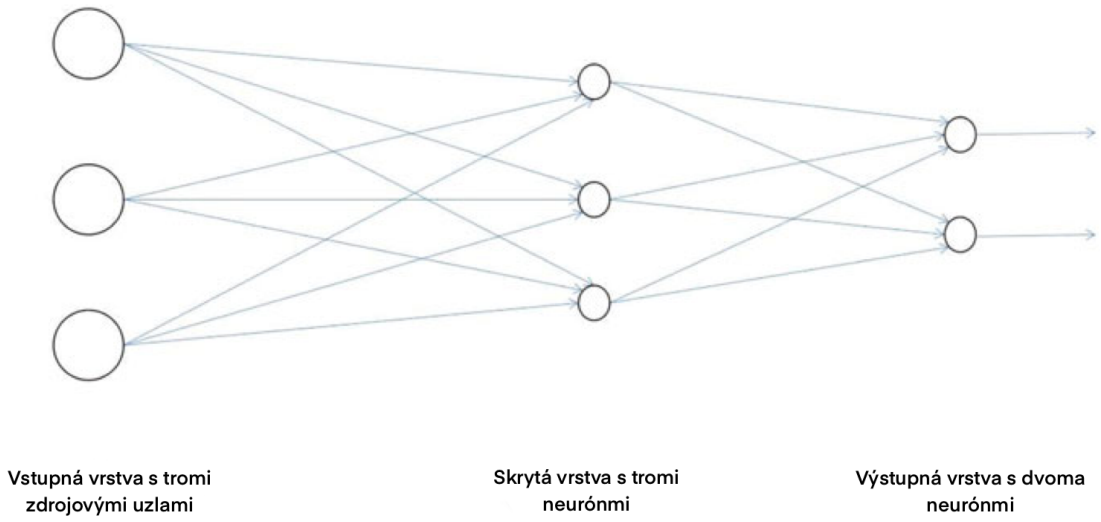
Aby sme mohli popísať konvolučnú neurónovú sieť, potrebujeme si najprv zadefinovať pojem neurónovej siete a popísať, z čoho sa skladá. NN je štruktúrou vzájomne prepojených jednotiek skladajúcich sa z veľkého počtu neurónov. Každý z týchto neurónov na vstupe prijíma vstupné dáta, ktoré v ďalších vrstvách spracuje a na výstupe ich odovzdáva ďalej ako výstup. Neuróny sú organizované do vrstiev a ak sa NN skladá len zo vstupnej a výstupnej vrstvy, hovoríme o jednovrstvovej NN (*angl. Single-layer Neural Network*). Túto sieť môžeme označiť aj ako jednovrstvový perceptrón (*angl. singlelayer perceptron*, SLP). NN však môže obsahovať aj jednu alebo viac skrytých vrstiev a v takom prípade hovoríme o hlbokej NN (*angl. Deep Neural Network*). Tá sa skladá z viacerých jednovrstvových perceptrónov, ktoré tvoria viacvrstvový perceptrón (*angl. multilayer perceptron*, MLP).

Jednovrstvová NN je najstaršia a zároveň najjednoduchšia umelá neurónová sieť (*angl. Artificial Neural Network*, ANN). Skladá sa zo vstupnej vrstvy zdrojových uzlov a výstupnej vrstvy neurónov viď. Obr.1 – 1. Zdrojové uzly sú priamo premietané na neuróny a celá ANN má len jednu vrstvu, nakoľko vstupná vrstva sa nezapočítava, pretože na nej neprebiehajú žiadne výpočty (Zhang, 2018).



Obrázok 1 – 1 Jednovrstvová NN

Hlboká NN sa skladá zo vstupnej vrstvy zdrojových uzlov, jednej alebo viacerých skrytých vrstiev a výstupnej vrstvy pozostávajúcej z neurónov viď. Obr.1 – 2.



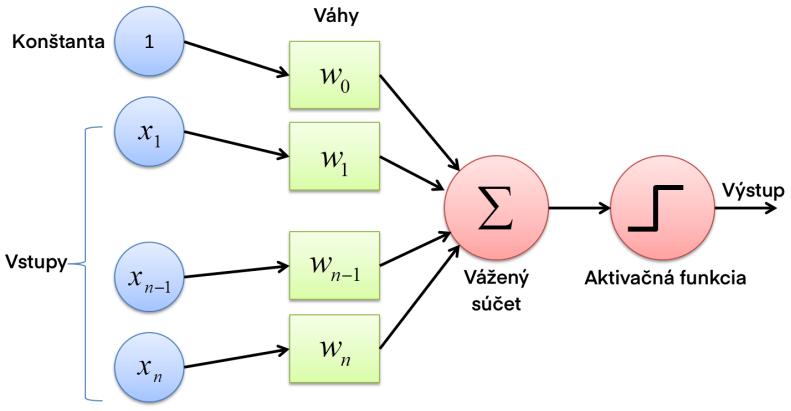
Obrázok 1 – 2 Hlboká NN

Takáto sieť je nazývaná aj ako dopredná neurónová sieť (*angl. feedforward neural network*, FNN) z dôvodu jednosmerného toku informácií cez sieť. Skryté vrstvy umožňujú NN extrahovať štatistické príznaky vyššieho rádu zo vstupu. Tieto vrstvy

nie sú priamo viditeľné pre vstupnú a výstupnú vrstvu. Neuróny nachádzajúce sa v skrytej vrstve nazývame skryté neuróny (*angl. hidden neurons*) a ich funkciou je zasahovať medzi vonkajším vstupom a výstupom siete. Zdrojové uzly vstupnej vrstvy dodajú prvky aktivačného vzoru, ktoré vytvoria vstupné signály aplikované na neuróny prvej skrytej vrstvy. Výstupný signál neurónov prvej vrstvy je následne vstupným signálov neurónov druhej vrstvy a takto to postupne pokračuje až kým sa signál nedostane na konečnú výstupnú vrstvu. Výstupné signály neurónov na výstupnej vrstve predstavujú celkovú odozvu siete na prvky aktivačného vzoru dodané zdrojovými uzlami vstupnej vrstvy (Zhang, 2018). Matematicky vieme operáciu perceptrónu neurónovej siete vyjadriť vzťahom 1.1,

$$y(t) = f(\sum_{i=0}^{m} w_i(t) * x_i(t) + b) \tag{1.1}$$

kde x je vstup, y výstup, w je váha a b je *bias*. Aktivačná funkcia je označená f a čas t (Gupta et al., 2013). Vizualizácia perceptrónu siete je znázornená na Obr. 1 – 3.



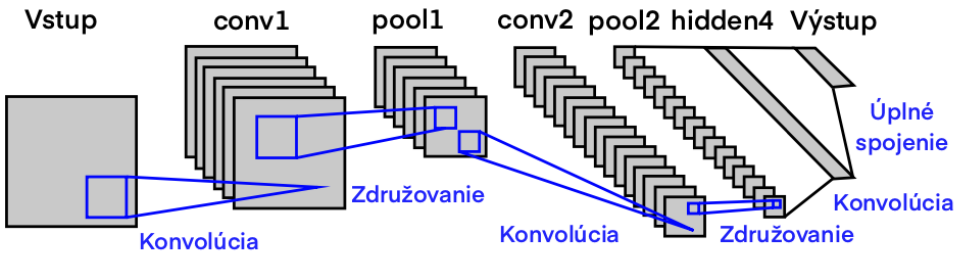
Obrázok 1 – 3 Perceptrón

Ak skryté vrstvy NN obsahujú matematickú operáciu konvolúcie, hovoríme o konvolučných neurónových sieťach. Operácia konvolúcie je operáciou dvoch matematických funkcií vyjadrená vzťahom 1.2.

$$s(t) = (x * w)(t) \tag{1.2}$$

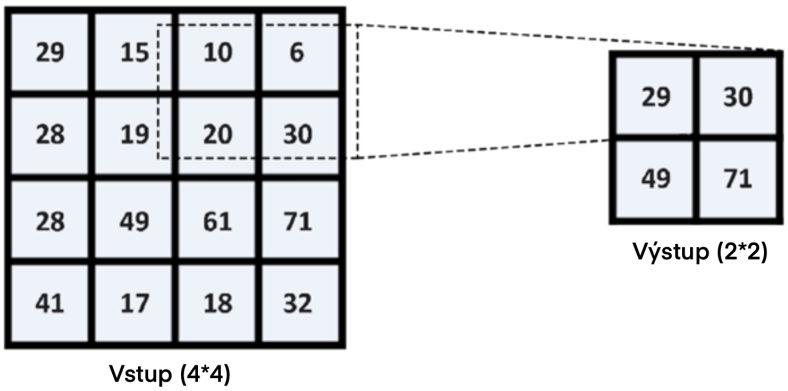
V tomto vzťahu sú w a x funkcie a znamiekno $*$ vyjadruje ich konvolúciu. Funkcia w musí byť valídna funkcia hustoty pravdepodobnosti a zároveň musí nadobúdať nulové hodnoty pre všetky negatívne argumenty. V terminológii CNN je argument x označovaný ako *input* siete a argument w je *kernel* siete. Výstup siete sa označuje ako mapa príznakov (*angl. feature map*) a t predstavuje časový údaj.

CNN spracúvajú dáta s topológiou založenou na mriežke. Dáta môžu byť mnohodimenzionálne ako napríklad obrazové dáta, alebo jednodimenzionálne, čoho príkladom môžu byť časové rady. CNN sa vyznačujú veľkou úspešnosťou v praktických aplikáciách. Ako už bolo spomenuté v úvode kapitoly 1, za ich úspechom stojí schopnosť vysporiadať sa s invariantnosťou v dátach. To znamená, že sa vedia vysporiadať s rôznou pozíciou hľadaného objektu, čo klasická NN nedokáže. Z Obr.1 – 4 vidíme, že sa vstupný obrázok postupne zmenšuje v skrytých vrstvách. Je to tým, že skryté vrstvy obsahujú operácie združovania (*angl. pooling*) (Goodfellow et al., 2016).



Obrázok 1 – 4 Vrstvy CNN

Združujúce vrstvy v CNN redukujú predchádzajúce vrstvy. Veľkosť poľa *poolingu* je vo väčšine prípadov nastavená na 2×2 a krok je nastavený na 1 alebo 2 aby sa zabránilo prekrývaniu. To znamená, že sa filter *max-poolingu* o veľkosti 2×2 posúva postupne vstupom s krokom veľkosti 1 alebo 2. Výstup *max-poolingu* bude mať veľkosť 2×2 viď. Obr.1 – 5.



Obrázok 1 – 5 Redukcia počtu pixelov združovaním

Pre zachovanie pôvodnej veľkosti výstupu je možné použiť metódu *padding*. Táto metóda doplní okraje výstupu nulami, čo môže zlepšiť extrakciu najmä tých príznakov, ktoré sú na okrajovej časti. Pri združovaní príznakov *poolingom* sa zachovávajú vzťahy medzi pixelmi. Vďaka tomu dokáže CNN rozoznať objekt, či už je preškálovaný, otočený alebo upravený iným spôsobom (Manaswi, 2018).

Ako už bolo spomenuté, výstup po operácii *max-pooling*, ktorý je zobrazený na Obr.1 – 5 je veľkosti 2×2. Výstup na Obr.1 – 4 je však jednorozmerný. Aby sme takýto výstup dosiahli, musíme ešte použiť operáciu *flatten*, v ktorej každý pixel rozvinieme do jednorozmernej reprezentácie (Ayyadevara, 2018). Výstup operácie *flatten* je zobrazený na Obr. 1 – 6.

Obrázok 1 – 6 Rozvinutie pixelov operáciou *flatten*

Teraz už máme prehľad o fungovaní konvolučných neurónových sietí. V nasledu-

júcich kapitolách načrtneme aké aktivačné, optimalizačné a regularizačné funkcie sa používajú v CNN. Následne popíšeme chybové funkcie a metódu spätného šírenia chýb. Teória NN je veľmi rozsiahla a preto sa budeme podrobne zameriavať iba na funkcie použité v praktickej časti práce. Pre podrobnejšie informácie odporúčame publikáciu *Deep Learning Book* od autorov Goodfellow et al. (2016).

Aktivačné funkcie

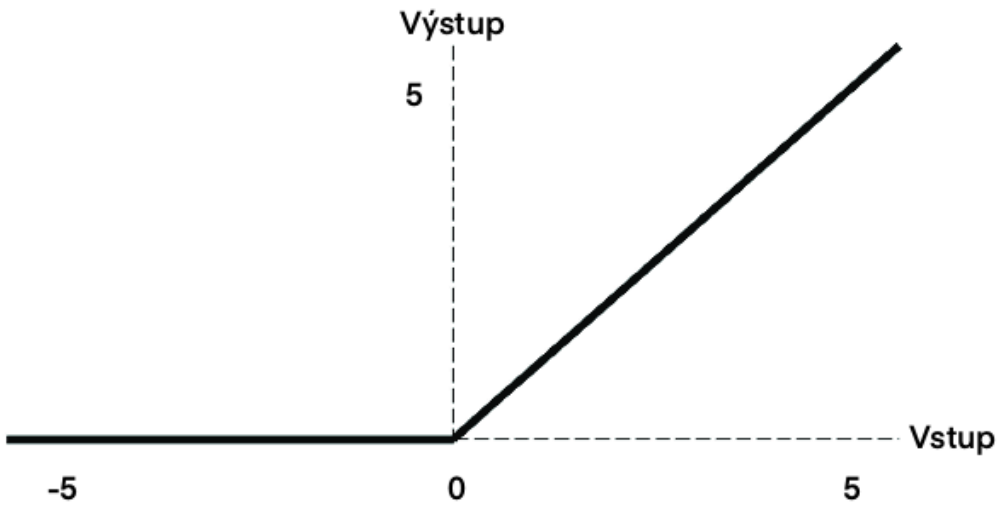
Aktivačné funkcie tvoria jadro každej štruktúry NN. Predstavujú rozhodovací element, ktorý definuje hranice vo vstupoch nastavením rozhodovacieho prahu. Bez aktivačných funkcií by výstupy boli len jednoduché lineárne funkcie. Aktivačné funkcie definujú to, ako sa vstup mení na výstup. Pri lineárnej funkcii sa neuróny správajú ako lineárny regresný model, takže je jedno koľko lineárnych funkcií za sebou použijeme, výstupom bude stále lineárna funkcia. Medzi základné aktivačné funkcie patria krokové a prahové funkcie. Tieto funkcie slúžia len ako binárne klasifikátory schopné poskytnúť iba pozitívny alebo negatívny výstup. Ak neuróny poskytujú iba tieto dva druhy výstupov, stráca sa veľa informácií o sile aktivácie. Pri neurónových sieťach sa používajú rozdielne aktivačné funkcie na skrytých vrstvách a na výstupnej vrstve. Použitie aktivačnej funkcie na výstupe vyberáme na základe riešenej úlohy. Ak riešime problém regresie, používame lineárnu aktivačnú funkciu. Pri binárnej klasifikácii by sme použili aktivačnú funkciu sigmoid a pri riešení problému viactriednej klasifikácie zase softmax aktivačnú funkciu (Santosh et al., 2022). V našej práci pri modelovaní používame na skrytých vrstvách ReLu funkciu a keďže riešime problém klasifikácie do viacerých tried, na výstupe používame aktivačnú funkciu softmax. Tieto funkcie dôkladne popíšeme v nasledujúcich podkapitolách.

Aktivačná funkcia ReLu

Rektifikovaná lineárna jednotka (*angl. Rectified Linear Unit*, ReLu) je takzvaná „piecewise“ funkcia. Funguje na princípe, že ak je hodnota vstupu menšia alebo

rovná nule, donúti výstup aby bol tiež nulový. V opačnom prípade je výstupná hodnota rovná vstupnej hodnote. Priame vynútenie nulových hodnôt na výstupe môže spôsobiť do určitej miery redšiu charakteristiku. Výhodou ReLu metódy je rýchlejší výpočtový výkon a neexistencia problému miznutia gradientov (*angl. Gradient vanishing*) pri vysokých hodnotách aktivácie. Priebeh ReLu funkcie je zobrazený na Obr. 1 – 7 a funkcia je definovaná vzťahom 1.3. ReLu metóda má aj svoje nevýhody. Keďže derivát ReLu funkcie je vždy nulový ak je vstupná hodnota negatívna, je funkcia náchylná na uviaznutie neurónov v stave, kedy sa nikdy neaktivujú a gradient na nich ostane nulový. Podstatou toho je, že keď je veľa takýchto uviaznutých neurónov, môže nastať problém takzvaných „miznúcich gradientov“, kedy sa učenie spomalí alebo v horšom prípade úplne zastaví (Wang et al., 2020).

$$f(x) = \max(0, x) \tag{1.3}$$



Obrázok 1 – 7 Priebeh ReLu funkcie

Existujú však malé obmeny tejto funkcie ako napríklad *leaky* ReLu, ktorá v negatívnej zóne narozdiel od klasickej ReLu funkcie povolí malý konštantný gradient a tým umožní váham zotaviť sa, ak je to potrebné (Santosh et al., 2022). Funkcia

leaky ReLu je definovaná vzťahom 1.4, kde α je malý konštantný gradient.

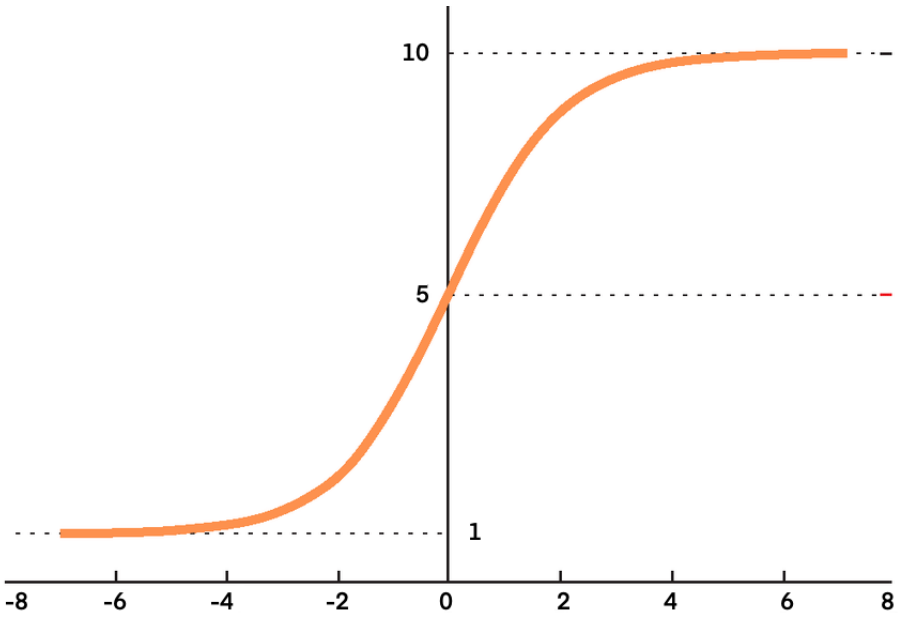
$$f(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha x & if\ x < 0 \end{cases} \tag{1.4}$$

Aktivačná funkcia Softmax

Softmax aktivačná funkcia predstavuje dôležitú funkciu, ktorá sa používa pri viactriednej klasifikácii v neurónových sieťach (Gao et al., 2020). Jedná sa o nelineárnu funkciu, ktorá predstavuje generalizáciu logistickej funkcie aplikovanej na viacero dimenzií. Vstup funkcie je vo forme vektora m reálnych čísel a tento vektor funkcia normalizuje na pravdepodobnostnú funkciu hustoty pozostávajúcu z m pravdepodobností (Haq and Jaffery, 2021). Táto funkcia môže vyžadovať vyššie hardvérové nároky kvôli jej zložitým exponentom a deliacim jednotkám. Funkcia je znázornená vzťahom 1.5,

$$f_n(\vec{x}) = \frac{e^{x_n}}{\Sigma_{m=1}^{M} e^{x_m}} \qquad n = 1, 2, \ldots . M \tag{1.5}$$

kde $x_1, x_2, ..., x_M$ sú vstupné hodnoty softmax vrstvy a výstupná hodnota $f(x_i)$ reprezentuje pravdepodobnosť, že daný príklad patrí do *i-tej* triedy (Wang et al., 2018). Suma výstupných hodnôt je vždy rovná jednej. Softmax sa väčšinou používa v poslednej vrstve NN, takže vstupy softmax vrstvy musia byť normalizované aby bolo možné distribúciu vstupných dát namapovať na vhodný rozsah spracovania. Pri preskočení kroku normalizácie by hrozila veľká strata presnosti modelu. Softmax funkcia môže byť použitá aj v skrytých vrstvách, napríklad pri vytváraní modelu, ktorý je navrhnutý aby vyberal jednu z m možností pre vnútornú premennú (Gao et al., 2020). Priebeh softmax funkcie je znázornený na Obr. 1 – 8.



Obrázok 1 – 8 Priebeh Softmax funkcie

Optimalizácia

Algoritmy strojového učenia zapájajú optimalizáciu v mnohých smeroch. Často sa používa vo forme analytickej optimalizácie, napríklad pri návrhu algoritmov. Zo všetkých optimalizačných problémov sú najzložitejšie práve tie, ktoré sú spojené s trénovaním neurónových sietí. Je celkom bežné vynaložiť dni až mesiace a zapojiť veľké množstvo strojov na vyriešenie čo i len jedného problému trénovania NN. Pre veľkú dôležitosť boli na vyriešenie tohoto problému vyvinuté rôzne špecializované techniky optimalizácie, ktoré sú rozdelené na klasické algoritmy, ako napríklad stochastický gradientový zostup (*angl. stochastic gradient descend*) a algoritmy s adaptívnym tempom učenia (*angl. algorithms with adaptive learning rates*). Medzi tieto algoritmy zaraďujeme napríklad algoritmus AdaGrad, RMSProp a Adam (Goodfellow et al., 2016). Práve posledný menovaný algoritmus využívame v našej práci.

Adam

Optimalizačný algoritmus Adam je metóda s priamočiarou implementáciou, ktorá je nenáročná na výpočtový výkon a je vhodná na riešenie problémov, ktoré obsahujú veľa dát alebo parametrov. Metóda je tiež vhodná pre nestacionárne ciele (*angl. non-stationary objectives*) a problémy s veľkým šumom alebo riedkym gradientom. Interpretácia hyperparametrov metódy je intuitívna a väčšinou sa vyžadujú len malé úpravy parametrov. Metóda funguje na princípe výpočtu individuálnych rýchlostí adaptívneho učenia pre rôzne parametre z odhadov prvého a druhého momentu gradientov.

Metóda využíva výhody populárnych metód AdaGrad a RMSProp. Pri AdaGrade je to schopnosť pracovať s riedkymi gradientmi a z RMSProp prebral dobré fungovanie v nestacionárnom prostredí. Veľkosť aktualizácie parametrov je invariantná voči zmene mierky gradientu, veľkosť kroku je ohraničená približne veľkosťou kroku hyperparametru a algoritmus prirodzene vykonáva formu krokového žíhania (*angl. step size annealing*) (Kingma and Ba, 2017). Fungovanie algoritmu Adam je popísané pseudo kódom 1.

Regularizácia

Častým riešeným problémom v strojovom učení je navrhnúť algoritmus, ktorý bude dosahovať dobré výsledky nie len na trénovacej dátovej množine ale aj na nových dátach, ktoré algoritmus nikdy nevidel. Mnoho použitých stratégií v ML je explicitne navrhnutých na minimalizáciu testovacej chyby (*angl. test error*) aj za cenu zvýšenia trénovacej chyby (*angl. train error*). Tieto stratégie sa súhrnne označujú regularizácia (*angl. regularization*).

V hlbokom učení (*angl. deep learning*, DL) je väčšina regularizačných stratégií založená na regularizácii estimátorov. Regularizácia estimátora funguje na hľadaní kompromisu medzi *bias* a odchýlkou (*angl. variance*). Dobrý regulátor učenia je ten, ktorý dokáže značne znížiť odchýlku a veľmi nenavýšiť *bias* (Goodfellow et al., 2016).

Algorithm 1: Algoritmus Adam

Požiadavka: Veľkosť kroku ϵ (predvolená hodnota je 0.001)

Požiadavka: Konštanta δ pre numerickú stabilizáciu (predvolená hodnota je 10^{-8})

Požiadavka: Exponenciálne miery poklesu pre momentové odhady ρ_1 a ρ_2 v ohraničení $[0, 1)$ (predvolená hodnota je 0.9 a 0.999)

Požiadavka: Počiatočné parametre θ

Inicializuj prvú a druhú momentovú premennú **s=0**, **r=0**

Inicializuj časový krok $t = 0$

while *ukončovacie kritérium nenaplnené* **do**

- Vytvor vzorku m príkladov z trénovacej množiny $\{x^{(1)}, ..., x^{(m)}\}$ s korešpondujúcim labelom $y^{(i)}$
- Vypočítaj gradient: $g \leftarrow \frac{1}{m} \nabla\theta \sum_i L(f(x^{(i)}; \theta), y^{(i)})$
- $t \leftarrow t + 1$
- Aktualizuj *biased* odhad prvého momentu: $s \leftarrow \rho_1 s + (1 - \rho_1)g$
- Aktualizuj *biased* odhad druhého momentu: $r \leftarrow \rho_2 r + (1 - \rho_2)g$
- *Correct bias* prvého momentu: $\hat{s} \frac{s}{1-\rho_1^t}$
- *Correct bias* druhého momentu: $\hat{r} \frac{r}{1-\rho_2^t}$
- Vypočítaj aktualizáciu: $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$
- Použi aktualizáciu: $\theta \leftarrow \theta + \Delta\theta$

end

V našej práci používame regularizačné stratégie augmentácie, včasného zastavenia trénovania (*angl. early stopping*) a dropout.

Augmentácia dát

Najlepším spôsobom ako dosiahnuť lepšie výsledky modelu je trénovať ho na väčšom počte dát. To však môže byť niekedy problémové. Dát nemusí byť k dispozícii dostatočné množstvo alebo anotovanie dátovej množiny je náročné a časovo nákladné.

V takomto prípade si vieme pomôcť vytvorením umelých trénovacích dát, ktoré pridáme do trénovacej množiny.

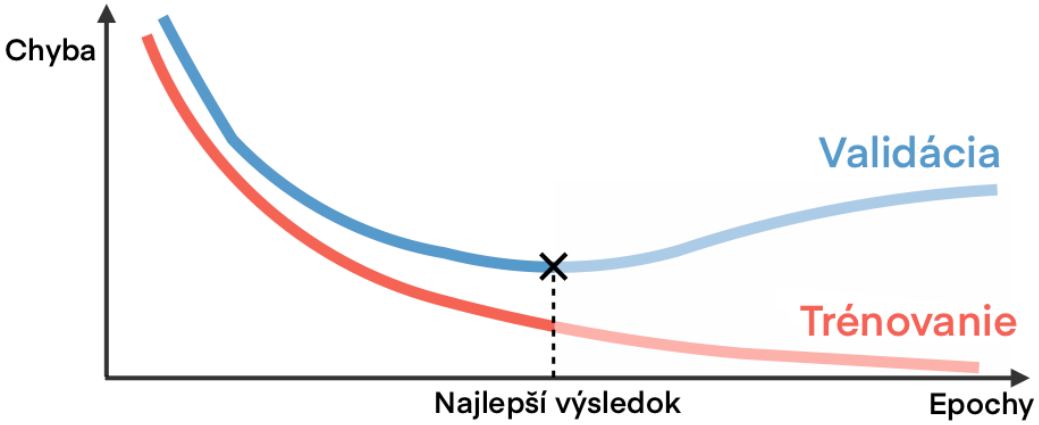
Najjednoduchšie sa tento prístup implementuje pri klasifikačných úlohách. Klasifikátor potrebuje sumarizovať mnohodimenzionálny vstup x jednou kategóriou y. Hlavným cieľom teda je aby bol klasifikátor nemenný vzhľadom na širokú škálu transformácií. Generovanie nových párov (x, y) dosiahneme jednoducho transformáciou vstupov x trénovacej množiny.

Táto technika dosahuje dobré výsledky hlavne pri špecifických klasifikačných problémoch ako je napríklad detekcia objektov (*angl. object recognition*). Obrázky sú mnohodimenzionálne a obsahujú veľa faktorov a variáci, ktoré môžu byť jednoducho simulované (Goodfellow et al., 2016). V našej práci sme sa zamerali na operácie, ktoré modifikujú kontrast, jas a gamu vstupných dát. Dôležité je použiť také operácie, ktoré nepretransformujú vstupy do podoby, v ktorej sa nevyskytujú v reálnych dátach. V našom prípade sme nepoužívali operácie ako posun, zmenšovanie a rotáciu vstupov, pretože v našich dátach sa objekty v takejto podobe nevyskytujú.

Predčasné ukončenie trénovania

Pri trénovaní modelu môžeme sledovať postupné znižovanie trénovacej chyby v priebehu trénovania ale validačná chyba začne znovu rásť. Tomuto javu sa hovorí preučenie modelu (*angl. overfitting*), kedy sa model až veľmi špecificky naučí črty trénovacej množiny a potom už nie je schopný efektívne rozoznať iné ako trénovacie dáta. Tomuto javu sa snažíme zabrániť aj stratégiou *early stopping*.

Na obrázku 1 – 9 vidíme, že chyba na trénovacej množine stále klesá, ale chyba pri validačnej množine po poklese opäť narastie. To znamená, že pri predčasnom ukončení trénovania vieme získať model s lepšou validačnou chybou a tým pádom aj lepším správaním na testovacej množine. Počas trénovania sa ukladá len model, ktorý má najlepší parameter validačnej chyby. Ak sa v nasledujúcej epoche trénovania validačná chyba nezlepší, najlepší uložený model sa neaktualizuje. Po skončení trénovania algoritmus vráti model s najlepšími parametrami z hľadiska validačnej



Obrázok 1 – 9 Priebeh trénovania modelu

chyby, namiesto modelu s parametrami na poslednej epoche. Vďaka svojej efektívnosti a jednoduchosti je technika *early stopping* jednou z najpoužívanejších techník v DL (Goodfellow et al., 2016).

Dropout

Dropout predstavuje výpočtovo nenáročnú ale výkonnú metódu regularizácie pri NN (Goodfellow et al., 2016). Opäť je to technika, ktorá sa snaží zabrániť preučeniu modelu a to takým spôsobom, že počas trénovania vypne neuróny spolu s ich väzbami v NN. To zabráni naučeniu veľmi špecifických príznakov a príliš veľkej adaptácii na trénovaciu množinu. Výber neurónov, ktoré sa majú vypnúť je náhodný a používateľ špecifikuje len hodnotu p, ktorá určuje pravdepodobnosť, koľko neurónov sa má vypnúť. Použitím dropoutu vznikajú stenčené NN (*angl. thinned neural network*). NN s n neurónmi môžme považovať za kolekciu 2^n možných stenčených NN. Tieto siete medzi sebou stále zdieľajú váhy, takže celkový počet parametrov je stále $O(n^2)$ alebo menej. Technika dropout použitá v NN je vyjadrená vzťahmi 1.6 a 1.7,

$$r_i^{(l)} \sim \text{Bernoulli}(p) \tag{1.6}$$

$$\tilde{\mathbf{y}}^{(l)} = \mathbf{r}^{(l)} * \mathbf{y}^{(l)} \tag{1.7}$$

$$\mathbf{z}^{(l+1)} = W^{(l+1)}\tilde{\mathbf{y}}^{l} + \mathbf{b}^{(l+1)} \tag{1.8}$$

$$\mathbf{y}^{(l+1)} = f(\mathbf{z}^{(l+1)}) \tag{1.9}$$

kde $\mathbf{r}^{(l)}$ je vektor Bernulliho (*angl. Bernoulli*) náhodných premenných, kde každá má pravdepodobnosť p, ktorá je 1. Tento vektor je združovaný pre každú vrstvu a znásobený výstupmi tejto vrstvy $\mathbf{y}^{(l)}$ aby sa vytvoril stenčený výstup $\tilde{\mathbf{y}}^{(l)}$. Stenčené výstupy sú následne použité ako vstupy ďalšej vrstvy. Vzťahy 1.8 a 1.9 predstavujú doprednú operáciu (*angl. feed forward operation*) NN, kde f je aktivačná funkcia. Počas fázy trénovania sú deriváty chybovej funkcie spätne šírené cez stenčenú sieť. Vo fáze testovania sú váhy škálované ako $W_{test}^{(l)} = pW^{(l)}$. Výsledná NN je spustená bez dropoutu (Srivastava and Srivastava, n.d.).

Chybová funkcia

Chybová funkcia (*angl. loss function*) v neurónových sieťach kvantifikuje rozdiel medzi očakávaným výsledkom a výsledkom vyprodukovaným modelom. Chybová funkcia počíta tento rozdiel na trénovacej množine dát a aktualizuje váhy modelu s cieľom minimalizácie chyby. Pre rôzne úlohy strojového učenia používame rôzne chybové funkcie. Výber chybovej funkcie taktiež priamo súvisí s použitou aktivačnou funkciou na výstupnej vrstve neurónovej siete. Nesprávny výber chybovej funkcie by viedol k neuspokojivým výsledkom modelu. Pri riešení problémov regresie je často využívaná priemerná kvadratická chyba (*angl. mean squared error*, MSE) a pri klasifikačných úlohách používame chybovú funkciu *Cross-Entropy* (Goodfellow et al., 2016). Túto chybovú funkciu delíme podľa toho, či riešime binárny klasifikačný problém alebo problém viactriednej klasifikácie. V našej práci realizujeme viactriednu klasifikáciu a preto sme použili chybovú funkciu *Categorical Cross-Entropy*.

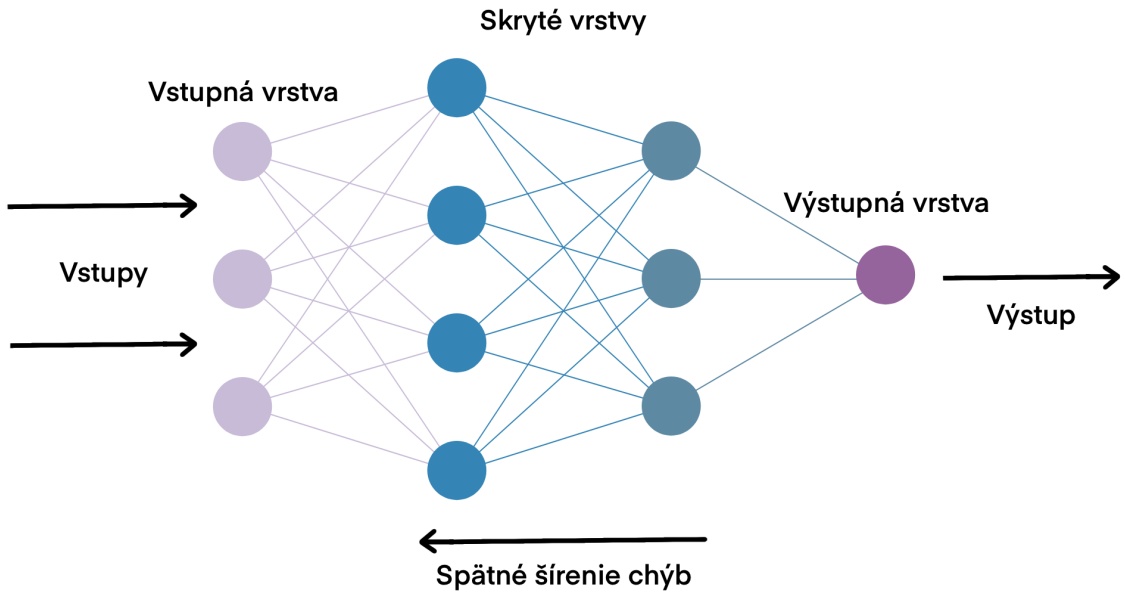
Táto funkcia násobí očakávaný výsledok logaritmom výsledku, ktorý bol vyprodukovaný modelom pre každú klasifikačnú triedu a následne ich spočíta. Výstupom je potom vektor, ktorého veľkosť sa rovná počtu klasifikačných tried modelu. Tento vektor obsahuje sumu pravdepodobností pre každú klasifikačnú triedu a súčet týchto súm je rovný jednej. Táto funkcia je definovaná vzťahom 1.10, kde M je počet kla-

sifikačných tried (Hu et al., 2018).

$$L = \sum_{j=1}^{M} y_j \log(\hat{y}_j) \tag{1.10}$$

Metóda spätného šírenia chýb

Základnou myšlienkou algoritmu spätného šírenia chýb (*angl. backpropagation*) je počítanie gradientu chybovej funkcie berúc v úvahu váhy. Algoritmus sa riadi reťazovým pravidlom a gradient počíta postupne po vrstvách. Algoritmus iteruje späť od poslednej vrstvy aby zabránil redundantným výpočtom. Funguje to takým spôsobom, že po každom prechode neurónovou sieťou vpred vykoná algoritmus spätný prechod a za spätného šírenia chýb aktualizuje váhy modelu. Týmto procesom optimalizuje hodnotu chybovej funkcie (Rumelhart et al., 1995). Fungovanie *backpropagation* algoritmu v NN je znázornené na Obr. 1 – 10.



Obrázok 1 – 10 Fungovanie algoritmu spätného šírenia chýb

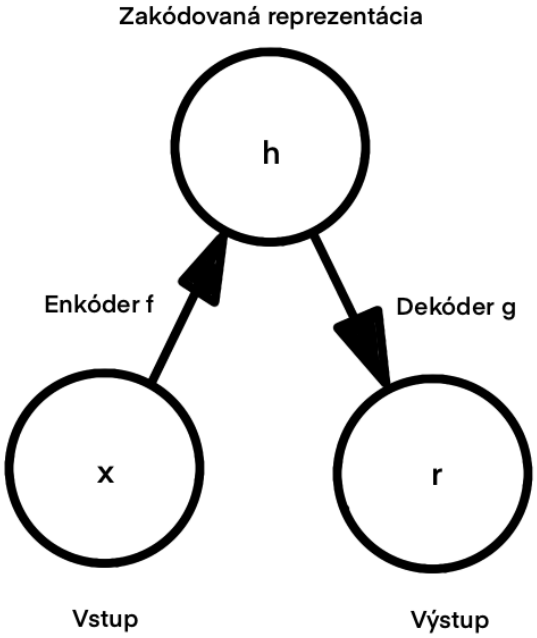
1.2 Nekontrolované učenie

Táto podkapitola je členená do dvoch častí podľa využitých metód v našej práci. Prvou časťou je použitie NN v podobe autoenkódera v kombinácii s metódou k-Means a druhá časť sa venuje metóde samo-organizujúcich sa máp. Nekontrolované učenie rieši problémy, v ktorých nemáme k dispozícii anotovanú dátovú množinu. Zaradenie príkladov do tried teda nie je vopred známe (Dougherty, 2013).

Fungovanie nekontrolovaného učenia môžeme popísať nasledujúcim príkladom. Predstavme si stroj alebo akýkoľvek žijúci organizmus, ktorý na začiatku dostane súbor vstupov $x_1, x_2, x_3, \ldots$, kde x_t je zmyslový vstup v čase t. Týmto vstupom sú naše dáta, ktoré môžu zodpovedať obrazu, ktorý vidíme, zvukovým vlnám alebo pixelom v kamere. Ako sme už načrtli na začiatku kapitoly 1, poznáme tri druhy ML. V našom príklade by pri kontrolovanom učení boli stroju poskytnuté aj informácie o zaradení príkladov do kategórií $y_1, y_2, \ldots, y_n$. Cieľom stroja v takomto prípade je naučiť sa produkovať správny výstup pri nových vstupoch. Pri semi-kontrolovanom učení stroj interaguje s prostredím a produkuje akcie $a_1, a_2, \ldots, a_n$. Tieto akcie ovplyvňujú stav prostredia, čo vedie k produkovaniu odmien $r_1, r_2, \ldots, r_n$. Cieľom stroja je naučiť sa konať tak aby maximalizoval budúce odmeny. A nakoniec sa dostávame k nekontrolovanému učeniu, ktoré dostane vstupy $x_1, x_2, \ldots$, ale nedostane informácie o zaradení do tried y, ani žiadnu formu odmien r z prostredia. Môže sa nám naskytnúť otázka, čo teda dokáže stroj urobiť ak nedostane žiadne informácie o dátach alebo z prostredia. Je však možné vytvoriť formálny rámec pre nekontrolované učenie založený na fakte, že cieľom algoritmu je vytvoriť reprezentácie vstupných dát, ktoré sa dajú použiť na predikovanie budúcich vstupov. O nekontrolovanom učení teda môžme uvažovať ako o metóde, ktorá hľadá vzory v dátach nad rámec čistého neštruktúrovaného šumu (Ghahramani, 2004).

k-Means + autoenkóder

Autoenkóder je typ neurónovej siete, ktorá je trénovaná za účelom zreplikovania vstupu siete na jej výstupe. Vo vnútri obsahuje skrytú vrstvu **h**, ktorá predstavuje zakódovanú reprezentáciu vstupu. Samotná NN sa skladá z dvoch častí. Prvou je funkcia enkódera $\mathbf{h} = f(\mathbf{x})$ a druhou časťou je dekóder $\mathbf{r} = g(\mathbf{h})$, ktorý vytvára rekonštrukciu vstupu. Táto architektúra je zobrazená na Obr. 1 – 11, kde sa vstup x mapuje na rekonštruovaný výstup r cez zakódovanú reprezentáciu h. Časť f je enkóder, ktorý mapuje x na h a g je dekóder mapujúci h na r.



Obrázok 1 – 11 Autoenkóder

Autoenkóder nie je navrhnutý aby vedel perfektne zreplikovať vstupy, ale aby sa naučil len ich približnú reprezentáciu. Väčšinou je nejakým spôsobom obmedzovaný, aby vedel vstupy kopírovať len približne a aby kopíroval iba vstupy, ktoré sa podobajú na trénovacie dáta. Z dôvodu, že model musí priorizovať, ktoré aspekty vstupov majú byť kopírované sa často naučí dôležité črty dát (Goodfellow et al., 2016). V našej práci sme autoenkóder použili na redukciu dimenzionality dát. Vstupné dáta

sme zakódovali a zhlukovanie pomocou k-Means už robíme na zakódovanej reprezentácii vstupných dát, ktorá obsahuje dôležité črty ale zároveň je menej náročná na spracovanie vďaka menšiemu počtu dimenzií.

Zhlukovanie k-Means je jednoduchá a rýchla metóda nekontrolovaného učenia, ktorá je numerická, nedeterministická a iteratívna. Používa sa v procesoch dolovania dát (*angl. data mining*) na zhlukovanie veľkych dátových súborov. V dátach hľadá vzory a na základe nájdených vzorov a rozdielov ich zoskupuje do nesúrodých zhlukov. Výsledkom sú zhluky, v ktorých sú si dáta podobné ale medzizhlukovo rozdielne. Jedná sa o deliaci zhlukovací algoritmus, ktorý klasifikuje dáta do k rozdielnych zhlukov iteratívnym konvergovaním k lokálnemu minimu.

Algoritmus pozostáva z dvoch častí, kde prvou je inicializácia náhodných centroidov, kde k je dopredu zafixovaná hodnota. V druhej fáze sa následne priradzujú dáta k najbližšiemu centroidu. Na odhad vzdialenosti medzi dátovými objektmi a centroidmi sa predvolene používa euklidovská vzdialenosť (*angl. Euclidean distance*). Po priradení všetkých dát do zhlukov sa následne znovu počítajú priemery vytvorených zhlukov. Tento proces je iteratívny a pokračuje až pokým sa kriteriálna funkcia nestane minimom. Kriteriálna funkcia je definovaná vzťahom 1.11,

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - x_i|^2 \tag{1.11}$$

kde E je suma kvadratickej chyby všetkých objektov v databáze, x je cieľový objekt a x_i je priemer zhluku C_i. Počet zhlukov je vopred definované k. Euklidovskú vzdialenosť $d(x_i, y_i)$ medzi dvoma vektormi x a y je možné získať funkciou 1.12.

$$d(x_i, y_i) = [\sum_{i=1}^{n} (x_i - y_i)^2]^{\frac{1}{2}} \tag{1.12}$$

Metóda k-Means má aj svoje nevýhody. Algoritmus musí vzdialenosti medzi dátami a centroidmi počítať v každej iterácii. To však nie je potrebné ak predpokladáme, že zhluk C je vytvorený po prvých j iteráciách a dátový objekt x je priradený zhluku C. Po ďalších iteráciách však objekt x aj naďalej ostáva priradený zhluku C,

pretože vzdialenosť ku centroidu tohoto zhluku je najmenšia. Vzdialenosti objektu x sa však počítajú aj ku ostatným centroidom zhlukov a to zaberá veľa času, čo v konečnom dôsledku ovplyvňuje efektívnosť algoritmu (Na et al., 2010).

SOM

Samo-organizujúce sa mapy patria do kategórie architektúry neurónových sietí, kde susedné bunky v NN súťažia vo svojich aktivitách prostredníctvom vzájomnej laterálnej interakcie. Bunky sa tak vyvinú do špecifických detektorov rôznych vzorcov signálu. Túto kategóriu môžeme nazvať aj kompetetívne, nekontrolované alebo samo-organizujúce sa učenie.

SOM je typ NN, kde sú bunky špecificky ladené na rôzne vzory vstupného signálnu pomocou procesu nekontrolovaného učenia. V základnej verzii aktívne odpovedá iba jedna bunka alebo iba skupina buniek na aktuálny vstup. Miesta odpovedí sa majú tendenciu usporadúvať v zmysluplnom smere. To znamená, že miesta blízko seba budú s najväčšou pravdepodobnosťou obsahovať objekty, ktoré sú si do určitej miery podobné alebo zdieľajú nejakú podobnú črtu. Objekty, ktoré sú v systéme koordinátov vzdialené budú naopak odlišné. Tento systém koordinátov si môžeme predstaviť ako dvojdimenzionálnu mriežku. Bunky umiestnené v mriežke budú teda predstavovať zhluky príkladov (Kohonen, 1990).

Výhodou SOM metódy je možnosť vizualizovať jej výsledky. To nám poskytuje možnosť empirického skúmania zhlukov. Navyše sa vieme dostať aj ku informáciám, aké vstupy sa nachádzajú v jednotlivých zhlukoch a aj na základe toho vyhodnotiť model. Zhluky môžeme analyzovať samostatne, alebo môžeme podobné zhluky na mape spojiť do väčších oblastí a analyzovať ich ako celok.

Zhlukovanie v SOM pozostáva z dvoch úrovní. V prvej sa zhlukujú dáta za použitia SOM a v druhej sa zhlukuje samotná SOM. To značne znižuje nároky na výpočtový výkon a umožňuje spracovanie objemnejších datasetov. Trénovanie SOM prebieha tak, že každá bunka i dvojdimenzionálnej mriežky je reprezentovaná prototypovým vektorom $\mathbf{m}_i = [m_{i1}, ..., m_{id}]$, kde d je vstupná vektorová dimenzia. Bunky

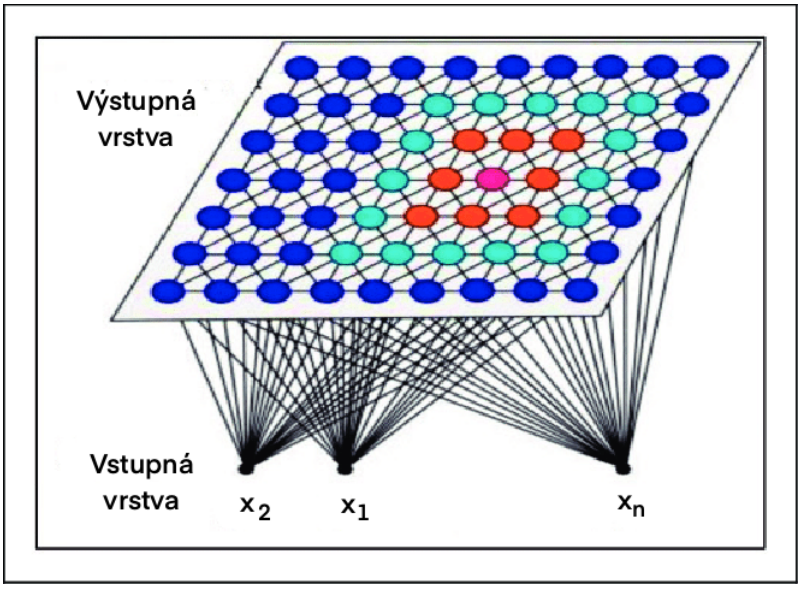
sú prepojené so susediacimi bunkami pomocou susedných vzťahov. SOM je trénovaná iteratívne a v každom trénovacom kroku sa zo vstupu náhodne vyberie vzorový vektor $\mathbf{x}$ a vypočítajú sa prototypové vektory m. Bunka mapy, ktorá obsahuje protyp najbližší ku $\mathbf{x}$ je označená ako najlepšie zodpovedajúca bunka (*angl. best matching unit*, BMU). Tento proces je vyjadrený vzťahom 1.13.

$$||\mathbf{x} - \mathbf{m}_b|| = min_i\{||\mathbf{x} - \mathbf{m}_i||\} \tag{1.13}$$

Následne sa aktualizujú prototypové vektory a BMU so svojimi susedmi je posunutá bližšie ku vstupnému vektoru. Aktualizované pravidlo pre prototypový vektor i je popísané vzťahom 1.14,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)h_{bi}(t)[\mathbf{x} - \mathbf{m}_i(t)] \tag{1.14}$$

kde t je čas, $\alpha(t)$ adaptačný koeficient a $h_{bi}(t)$ susedné jadro sústredené na víťaznú bunku (Vesanto and Alhoniemi, 2000). Obr. 1 – 12 popisuje ako vyzerá mapovanie vstupov na výstupnú vrstvu, s BMU označenou fialovou farbou.



Obrázok 1 – 12 SOM mapovanie

Po natrénovaní sa následne mapa sama usporiada a vykoná sa krok extrakcie

zhlukov, kde sa jednotlivé zhluky identifikujú. Väčšinou sa to robí priamym skúmaním zhlukov. Nevýhodou tohto prístupu je fakt, že SOM metóda zvykne vstupy distribuovať pomerne rovnomerne, čo môže predstavovať problém pri datasetoch malých rozmerov alebo datasetoch obsahujúcich šumy. To je možné vyriešiť vyhladzovaním, po ktorom budú jednotlivé zhluky a hranice medzi nimi viditeľné (Brett et al., 2004). V našej práci využívame som metódu trochu odlišným spôsobom. Na mapu neprenášame žiadne označenia tried, ale mapa obsahuje samotnú reprezentáciu dát. Bližšie tento prístup popíšeme v podkapitole 5.3.

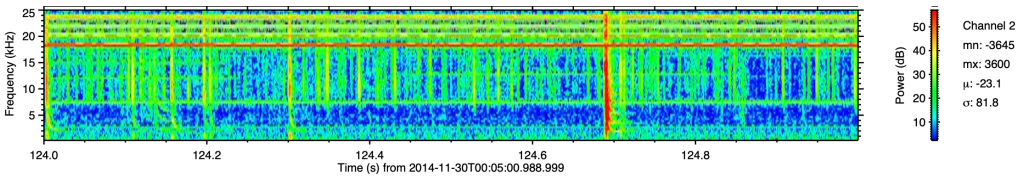
2 Analýza rádiových spektrogramov

Bleskové výboje je možné sledovať rádiovými prijímačmi, ktoré slúžia na meranie elektromagnetických pulzov vo veľmi nízkom frekvenčnom rozsahu (*angl. very low frequency range*, VLF). VLF signály generované bleskovými výbojmi sa volajú atmosferiky (*angl. atmospherics*, sferik) a dokážu sa šíriť vzdialenosťou viac ako tisíc kilometrov, pričom sa odrážajú od vlnovodu Zem-ionosféra (*angl. Earth-ionosphere waveguide*), ktorý je tvorený vodivou zemskou plochou alebo oceánom na jednej strane a spodnou časťou vodivej ionosféry na druhej strane. Analýzou týchto signálov je možné získať informácie o búrkovej činnosti, predpovedať možné živelné katastrofy ako tornáda alebo hurikány a v neposlednom rade odhadnúť množstvo vyprodukovaných bleskov.

Horná hranica vlnovodu pre šírenie VLF signálu sa nazýva D-región a je v nadmorskej výške 50 až 90 kilometrov. V tejto výške je ovplyvňovaná Slnkom, takže merania za dňa a noci vykazujú odlišné vlastnosti z dôvodu pôsobenia slnečného žiarenia. Spodná hranica D-regiónu zase môže byť ovplyvnená silnými meteorologickými vplyvmi alebo seizmickou činnosťou. Ako už bolo spomenuté v úvode práce, elektromagnetické výboje boli namerané prístrojom ELMAVAN-G a v našej práci s nimi pracujeme vo forme časovo-frekvenčných spektrogramov vypočítaných pomocou troj-komponentovej analýzy.

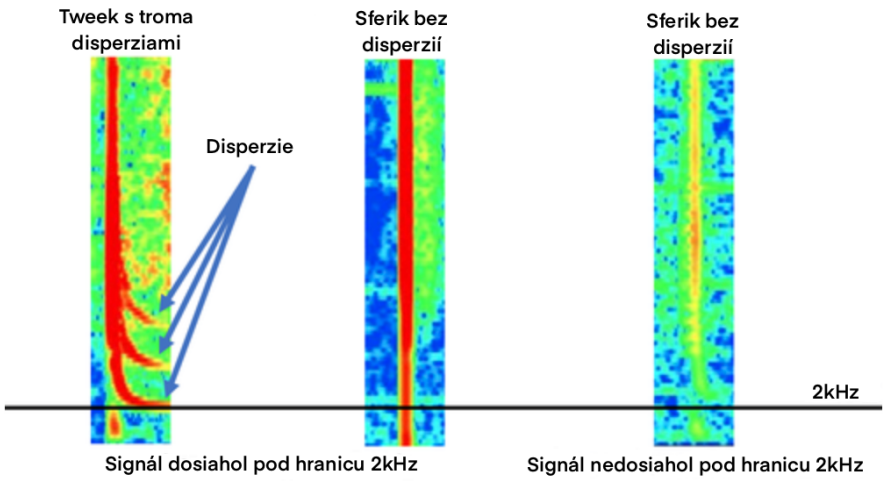
Prístroj ELMAVAN-G sa skladá z rádiového prijímača, ktorý je doplnený dvoma kolmými anténami s magnetickou slučkou a sférickou elektrickou anténou lokalizovanou 2 metre nad zemou. Prístroj bol vyvinutý Ústavom fyziky atmosféry Akademie věd České republiky a je lokalizovaný na vrchole La Grande Montagne blízko mesta Rustrel vo Francúzsku. Frekvenčný rozsah merania je $0 - 25kHz$ a trvanie merania je 144 sekúnd. Meracia sekvencia sa opakuje každých 5 minút. Z merania sa generuje 9 spektrogramov v trvaní jednej sekundy, kde bola elektromagnetická emisia najsilnejšia. Takýto spektrogram je zobrazený na Obr. 2 – 1, kde horizontálna os obsahuje časový údaj a vertikálna je frekvenčné rozpätie. Spektrogram ďalej obsa-

huje údaj o sile nameraných pulzov vyjadrených farebnou škálou v decibeloch. Na Obr. 2 – 1 môžeme vidieť aj horizontálnu čiaru prechádzajúcu celým spektrogramom vo frekvenčnom pásme $16 - 18kHz$. Táto čiara je spôsobená elektromagnetickými signálmi emitovanými silnými VLF vysielačmi používanými na komunikáciu s ponorkami (Maslej-Krešňáková et al., 2021).



Obrázok 2 – 1 Spektrogram obsahujúci namerané udalosti prístrojom ELMAVAN-G

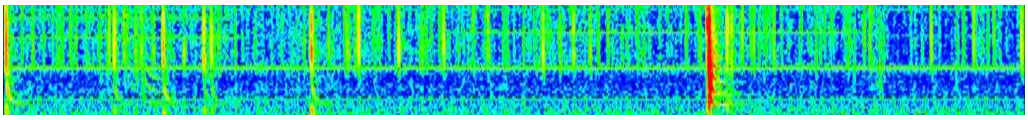
Podľa teórie vlnovodu, ktorú definoval autor Budden (1961), signály generované bleskovou činnosťou sa prenášajú vlnovodom rôznymi postupmi. Postup nultého rádu (*angl. zero-order mode*) sa šíri vlnovodom všetkými frekvenciami. Postupy vyššieho rádu (*angl. higher-order modes*) sa nemôžu šíriť pod ich kritické frekvencie. Sferiky, ktoré boli namerané v noci navyše vykazujú frekvenčné disperzie, ktoré v reproduktoroch produkujú špecifický zvuk takzvaného klikania. Tu sa už dostávame ku hlavnému deleniu nameraných signálov zachytených na spektrogramoch.



Obrázok 2 – 2 Delenie zachytených signálov na spektrogramoch

Na Obr. 2 – 2 sú zobrazené dve hlavné delenia zachytených udalostí. Vyššie spomínané sferiky, ktoré vykazujú frekvenčné disperzie, nazývame tweeky. Počet ich disperzií sa môže líšiť od jednej až po viac ako desať disperzií v jednom zachytenom signále. Druhou hlavnou triedou nášho záujmu sú signály, ktoré nevykazujú disperzie a nazývame ich jednoducho sferiky. Signály sa následne ešte líšia tým, či ich emisia siaha pod frekvenčnú hranicu $2kHz$.

Obr. 2 – 1 však nepredstavuje konečnú podobu spektrogramu, s ktorou pracujeme v našej práci. Pripravený spektrogram, ktorý používame v našej práci predstavuje Obr. 2 – 3. Zo spektrogramu boli odstránené informácie nepotrebné pre študovanú úlohu, ako napríklad horizontálny signál používaný pre komunikáciu s ponorkami, ktorý s našou prácou nijako nesúvisí. Takto upravený spektrogram je vhodný na ďalšie spracovanie.



Obrázok 2 – 3 Predspracovaný spektrogram vhodný ako vstup detekčného modelu

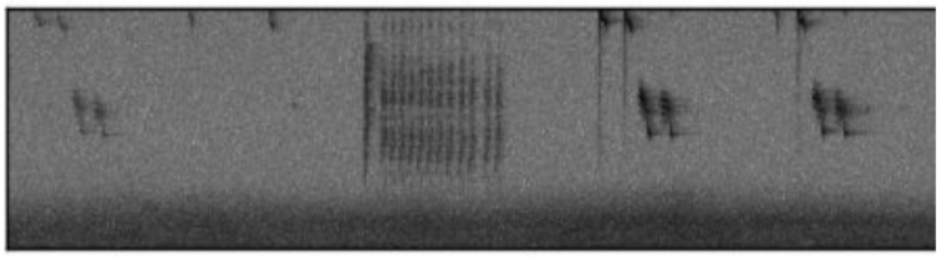
Takto nameraných dát máme obrovské množstvo. V priemere na zem udrie okolo 50 bleskov za sekundu, takže manuálna analýza týchto dát nepripadá v úvahu. Naskytá sa nám otázka, čo ďalej robiť s týmito dátami. Odpoveďou sú techniky strojového učenia, ktoré dokážu automaticky spracovať obrovské množstvo dát. V nasledujúcej kapitole si povieme o už existujúcich riešeniach a popíšeme náš doterajší výskum s jeho ďalším smerovaním.

3 Analýza súčasného stavu

Ako bolo spomenuté v úvode, táto práca nadväzuje na existujúci výskum Maslej-Krešňáková et al. (2021), ktorým som spoluautorom a je stručne popísaný v podkapitole 3.4. Okrem nášho výskumu existuje ešte niekoľko prístupov ku analýze spektrogramov alebo výskumov, ktoré sa venujú využitiu metód použitých aj v tejto práci. V nasledujúcich podkapitolách 3.1 a 3.2 popíšeme analýzu spektrogramov metódami strojového učenia a podkapitola 3.3 sa venuje využitiu metódy samo-organizujúcich sa máp v astro doméne na klasifikáciu astronomických svetelných kriviek.

3.1 Hlboké učenie použité na audio spektrogramoch

V práci Fanioudakis and Potamitis (2017) sa autori zamerali na detekciu a segmentáciu vokalizácie vtákov, ktorá bola nahratá v otvorenom poli. Automatická detekcia vtáčích zvukov môže byť použitá na monitorovanie rôznych druhov vtáctva. V práci boli tieto úlohy riešené dvoma prístupmi, kde jedným z nich bolo využitie frameworku YOLOv2 založenom na hlbokom učení. Tento framework dokáže lokalizovať a označovať objekty na vstupných dátach. Príkladom takýchto vstupov pre YOLO metódu je Obr. 3 – 1.

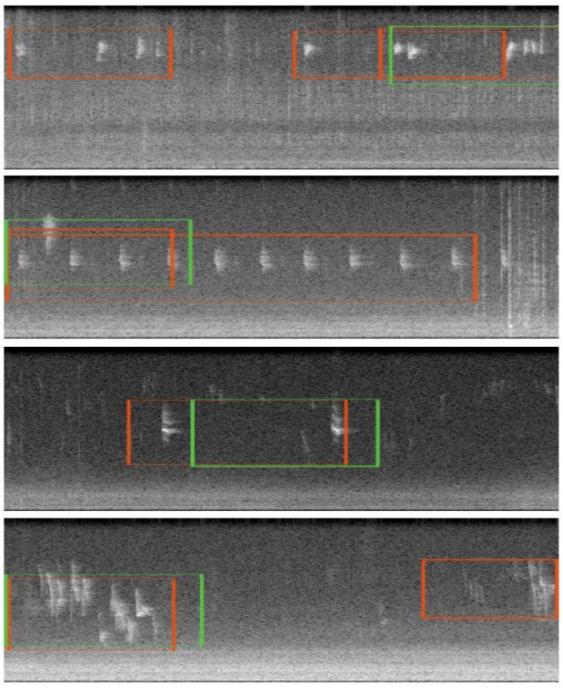


Obrázok 3 – 1 Spektrogram vokalizácie vtákov (Fanioudakis and Potamitis, 2017)

Názov tejto metódy *You Only Look Once* nám napovedá o jej spôsobe fungovania. YOLO zoberie celý vstupný obrázok v jednej inštancii a predikuje na ňom súradnice boxov ohraničujúcich objekty s pravdepodobnosťou ich výskytu. Pre tento prístup je YOLO veľmi rýchla metóda, čo predstavuje výhodu pri spracovaní veľkého množstva

dát.

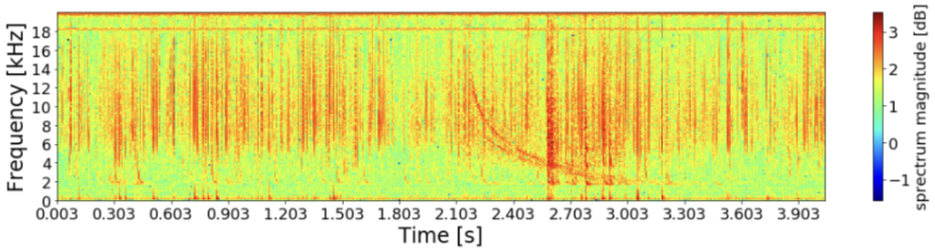
Práca ďalej popisuje rôzne vzory volania vtákov a ako sa z nich následne vytvorili spektrogramy. Tieto spektrogramy bolo následne potrebné anotovať a keďže to by bolo príliš časovo náročné, rozhodli sa autori anotovať len to, či spektrogram obsahuje volanie alebo ho neobsahuje. Následne anotovanú množinu použili ako vstup YOLO frameworku, ktorého úlohou bolo označiť volania na spektrograme a zaklasifikovať ich. Dáta, ktoré boli v práci využité sa skladali z viac ako 17000 audio nahrávok s dĺžkou 10 sekúnd. Všetky boli ručne anotované, či sa v nich nachádza alebo nenachádza vtáčí zvuk. Trénovanie modelu prebiehalo na 50 epochách s predtrénovanými váhami. Takýto model dosiahol v najlepšej konfigurácii presnosť 88.94% a návratnosť 94.76%, čo sú veľmi dobré výsledky. Tým autori dokázali použiteľnosť YOLO metódy na detekciu objektov nachádzajúcich sa na spektrogramoch. Obr. 3 – 2 predstavuje výstup YOLO metódy, kde môžeme vidieť vizuálne označené udalostí rôznofarebnými boxmi. Rôzna farba značí klasifikáciu do rôznych tried.



Obrázok 3 – 2 Nájdené udalosti metódou YOLO na spektrogramoch (Fanioudakis and Potamitis, 2017)

3.2 ML techniky na detekciu a charakterizáciu rádiových vĺn

Ďalším článkom podporujúcim využiteľnosť techník strojového učenia na analýzu spektrogramov je článok autorov Konan et al. (2020). Autori v tomto článku tiež skúmajú spektrogramy elektromagnetických pulzov vo VLF pásme, ktoré boli vytvorené bleskami a navyše ako môžeme vidieť na Obr. 3 – 3, dáta samotné sú podobné dátam, s ktorými pracujeme v našej práci. Práca sa však venuje detekcii udalostí hvizdov (*angl. whistler radio waves*), ktoré boli pozorované v plazmosfére.



Obrázok 3 – 3 Spektrogram hvizdov (Konan et al., 2020)

V práci sa spomína už existujúca metóda na automatickú detekciu hvizdov, ktorá bola vytvorená vedcom Lichtenbergrom v roku 2009. Táto metóda je založená na korelácii a vyžaduje si značný výpočtový výkon. Cieľom práce bolo vytvorenie ML modelu, ktorý by bol schopný automatickej detekcie hvizdov na spektrogramoch. Prístupom autorov bola klasifikácia obrázkov spojená s ich lokalizáciou. Dataset pozostával z približne 2300 udalostí identifikovaných metódou vytvorenou Lichtenbergrom. Autori navrhli tri prístupy pre riešenie tejto úlohy a to metódu podobnú tej Lichtenbergrovej, klasifikáciu obrázkov na oblastiach záujmu extrahovaných zo spektrogramu a nakoniec prístup využívajúci YOLO framework. Opäť sa zameriame práve na výsledky YOLO metódy, nakoľko sa táto metóda osvedčila aj vo vyššie spomenutom článku 3.1. Rozdiel bol však v použitej verzii YOLO frameworku, pretože v práci bola použitá vyššia verzia YOLOv3. Autori použili dva predtrénované modely, ktoré YOLO poskytuje a to YOLOv3-tiny a YOLOv3-spp, kde spp model vyšiel

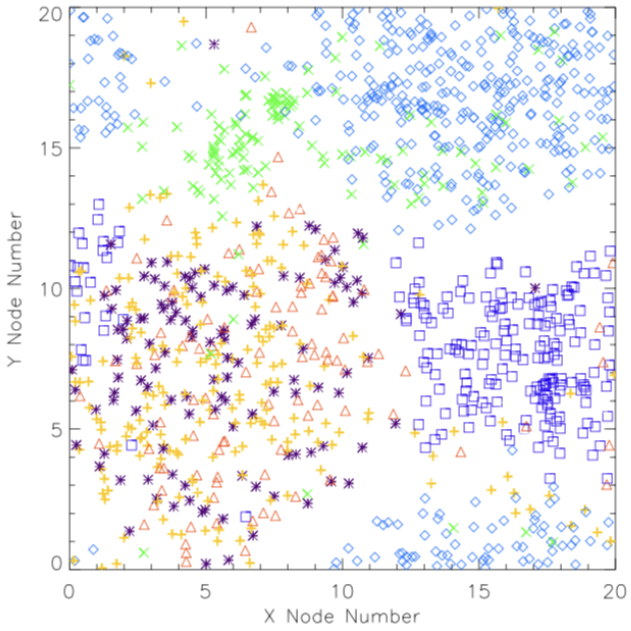
ako presnejší ale za cenu výpočtovej náročnosti, pretože bol 26 krát viac výpočtovo náročnejší ako tiny model. Autori v práci uviedli aj nevýhody YOLO modelu, medzi ktoré radia závislosť trénovania na anotácii trénovacej dátovej množiny, čo môže predstavovať problém z časového hľadiska. Ako ďalšiu nevýhodu uviedli problematickosť rozlišovania objektov, ktoré boli veľmi blízko seba. Aj napriek týmto problémom sa YOLO model ukázal ako schopný správne klasifikovať a nachádzať hvizdy, čo pre podobnosť vo vstupných dátach s našou prácou predstavuje veľmi dobrú správu.

3.3 Klasifikácia pomocou samo-organizujúcich sa máp

V predchádzajúcich článkoch 3.1 a 3.2 sme dokázali využiteľnosť YOLO frameworku na automatickú klasifikáciu udalostí nachádzajúcich sa na spektrogramoch. YOLO je však metóda využívajúca kontrolované učenie. Hoci ho v našej práci využívame na prácu s dátami, nepredstavuje náš hlavný predmet záujmu. Môžeme ho však použiť ako základný nástroj na spracovanie spektrogramov do vhodnejšej podoby pre ďalšiu analýzu metódami nekontrolovaného učenia.

Využitiu metódy nekontrolovaného učenia samo-organizujúcich sa máp sa venujú autori Brett et al. (2004) v článku o automatickej klasifikácii astronomických svetelných kriviek pomocou SOM. V práci bol algoritmus použitý na syntetických aj reálnych dátach a zistilo sa, že dokáže rozlišovať typy svetelných kriviek v oboch prípadoch. Syntetická dátová množina sa skladala z 5000 vygenerovaných svetelných kriviek s tvarovou distribúciou vykreslenou zo štyroch rozdielnych tried a reálne dáta sa skladali z 1206 svetelných kriviek, ktoré pochádzali z ROTSE experimentu (Akerlof et al., 2003). Prezentované testy boli získané na dvojdimenzionálnej sieti, ktorá poskytuje dostatočnú slobodu na sformovanie zhlukov. Navyše takto získané výsledky sú jednoduché na vizualizovanie a interpretáciu. Autori zistili, že veľkosť siete nemá zásadný vplyv na schopnosť formovania zhlukov. Vyskúšali veľkosti od 5×5 až po sieť veľkú 60×60. Dôležité však bolo zabezpečenie dostupnosti dostatoč-

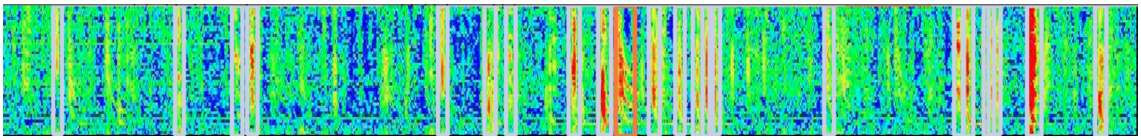
ného počtu neurónov, aby bola sieť schopná rozoznávať malé rozdiely vo svetelných krivkách, pričom bolo potrebné sa vyhnúť situácii, kedy by bol počet neurónov rovnaký alebo väčší ako počet členov v trénovacích dátach. Obr. 3 – 4 predstavuje SOM naučenú na 1206 predklasifikovaných ROTSE svetelných krivkách. Rôzne tvary na mape znázorňujú zhluky obsahujúce rôzne typy udalostí. V závere práce bola SOM metóda klasifikácie astronomických svetelných kriviek vyhodnotená ako schopná a spoľahlivá pre klasifikáciu syntetických aj reálnych kriviek. Ďalej bolo zistené, že použitá metóda je robustná a nie je veľmi závislá od parametrov kontrolujúcich učiaci proces. SOM metóda sa osvedčila ako cenný doplnok procesu dolovania v dátach pre analýzu veľkých a početných datasetov.



Obrázok 3 – 4 Vizualizovaná SOM mapa (Brett et al., 2004)

3.4 Automatická detekcia atmosferikov v rádiových spektrogramoch založená na metódach hlbokého učenia

V našom doposiaľ realizovanom výskume Maslej-Krešňáková et al. (2021) sme sa zamerali na automatickú detekciu udalostí vo frekvenčno-časových spektrogramoch. Detekcia bola realizovaná rovnako ako v článkoch 3.1 a 3.2 metódou YOLO, ktorá je založená na architektúre hlbokých neurónových sietí. V našom výskume sme však použili dosiaľ najaktuálnejšiu verziu YOLOv5. Hlavnou výhodou tejto metódy je, že po natrénovaní modelu na trénovacom datasete je algoritmus schopný detegovať objekty na vstupných dátach v reálnom čase a nepotrebuje pre detekciu reťazenie žiadnych iných procesov. Model, ktorý sme použili na detekciu sa volá YOLOv5s a skladá sa zo 7,3 milióna parametrov a 224 vrstiev. Natrénovaný bol na anotovanej množine skladajúcej sa z 2300 obrázkov, na ktorých sa nachádzalo 19000 udalostí typu sferik a 3200 udalostí typu tweek. Model bol trénovaný na 400 epochách s hyperparametrom *batch size* o veľkosti 4 a vstupné obrázky mali veľkosť 1184×160 pixelov. Výstupom modelu sú obrázky s ohraničujúcimi boxami, ktoré označujú jednotlivé typy udalostí. Príkladom takéhoto výstupu je Obr. 3 – 5. Okrem vizuálneho označenia udalostí, ukladá YOLO koordináty ohraničujúcich boxov spolu s typom udalosti do textových súborov a práve tie využívame v našom ďalšom výskume.



Obrázok 3 – 5 YOLO detekcia udalostí na spektrograme

Predikcie YOLO modelu sme využili na vytvorenie výstupnej tabuľky, ktorá obsahuje základné informácie o zachytených udalostiach. Výstupná tabuľka obsahuje stĺpce:

1. **image** – reprezentuje id anotovaného spektrogramu
2. **event** – poradové číslo lokalizovanej udalosti na spektrograme

3. **date** – extrahovaný dátum a čas vo formáte YYYYMMDD_HHMMSS
4. **second** – sekunda zachytenia udalosti v 144 sekundovom meraní
5. **milisecond** – presná milisekunda, kedy bola udalosť zachytená
6. **tweek** – binárna hodnota, kde 0 predstavuje udalosť typu sferik a 1 udalosť typu tweek
7. **f_min<2kHz** – binárna hodnota, kde 0 je udalosť, ktorej emisia je nad hranicou $2kHz$ a 1 je emisia pod hranicou $2kHz$

Na tomto výskume som sa podieľal formou vývoja softvéru, formálnou analýzou, prieskumom a taktiež som sa podieľal na vizualizácii a validácii výsledkov. Ďalšiemu výskumu, ktorému sa budeme v nasledujúcej kapitole 4 venovať, je vytvorenie modelu na báze NN, ktorý z YOLO výstupu dokáže zaklasifikovať udalosti typu tweek podľa počtu ich disperzií. Výstup vytvoreného modelu následne spojíme s výstupnou tabuľkou a rozšírime ju o stĺpec obsahujúci informáciu o počte disperzií. V kapitole 5 sa budeme venovať použitiu metód nekontrolovaného učenia k-Means a SOM s cieľom objaviť v dátach nové a zaujímavé vzory.

4 Klasifikácia disperzií

V tejto kapitole sa venujeme vytvoreniu klasifikačného modelu na báze konvolučných neurónových sietí, ktorý klasifikuje udalosti typu tweek podľa počtu disperzií. Cieľom je rozšírenie už existujúceho výskumu, realizovaného v článku Maslej-Krešňáková et al. (2021) o vytvorený model, ktorý prinesie nové informácie o zachytených udalostiach. Výsledky modelu spojíme s existujúcim výskumom a vytvoríme ucelený postup na klasifikáciu atmosferikov od ich stiahnutia a predspracovania, až po detekciu udalostí a interpretáciu výsledkov. Vytvoreným postupom bude následne možné analyzovať ktorýkoľvek rok nameraných dát spustením série krokov. V podkapitole 4.1 popíšeme všetky potrebné kroky a postupy, ktoré sme využili pri predspracovaní dátovej množiny do podoby vhodnej ako vstup pre trénovanie modelu. Podkapitola 4.2 popisuje štruktúru vytvoreného modelu a podkapitola 4.3 vyhodnotenie výsledkov na testovacej množine dát. Nakoniec popíšeme v podkapitole 4.4 postup spojenia modelu s predchádzajúcim výskumom a nasadenie celkového výsledku na nameraných dátach prístrojom ELMAVAN-G za jeden rok. Celkovo sme natrénovali tri modely pre klasifikáciu disperzií. Každý model bol natrénovaný na dátovej množine predspracovanej rozdielnym spôsobom. Jednotlivé spôsoby predspracovania sú popísané v podkapitole 4.1.

- **Model 1**

  Tento model má na vstupe dáta, ktoré boli predspracované spôsobom normalizácie.

- **Model 2**

  Tento model má na vstupe dáta, ktoré boli predspracované spôsobom vytiahnutia červeného kanálu.

- **Model 3**

  Tento model má na vstupe augmentované dáta.

4.1 Príprava dát

Nakoľko model klasifikuje udalosti typu tweek, je potrebné tieto udalosti získať a extrahovať zo spektrogramov pre potreby trénovania modelu. Na to sme využili anotačný projekt realizovaný na platforme Zooniverse (Smith et al., 2013). Tento projekt bol realizovaný už pri predchádzajúcom výskume Maslej-Krešňáková et al. (2021) a je popísaný v bakalárskej práci Kundrát (2021). Dostupnosť anotovanej dátovej množiny nám ušetrila veľké množstvo času. Výstupom projektu bol JSON súbor obsahujúci informácie o anotovaných udalostiach na spektrogramoch. JSON súbor však obsahoval okrem dôležitých informácii pre extrakciu aj veľa metadát, ktoré pre nás neboli potrebné. Súbor sme preto upravili tak, aby obsahoval len potrebné informácie a následne sme vyfiltrovali udalosti typu tweek a uložili ich do samostatného csv súboru. Takto upravená tabuľka obsahovala nasledujúce atribúty:

- **image** – názov obrázku
- **xmin** – *xmin* súradnica na spektrograme
- **ymin** – *ymin* súradnica na spektrograme
- **xmax** – *xmax* súradnica na spektrograme
- **ymax** – *ymax* súradnica na spektrograme
- **label** – 1 pre tweek
- **tweek_sign** – počet disperzií (1 až 9)

Analýzou datasetu sme zistili nerovnomerné zastúpenie udalostí z hľadiska počtu disperzií. Ako vidíme na tabuľke 4 – 1, udalosti s počtom disperzií 4 a viac sú oveľa menej početne zastúpené, ako udalosti s počtom 1, 2 a 3 disperzie. Preto sme sa rozhodli menej početné triedy spojiť a vytvoriť z nich jednu triedu. Výsledné triedy predstavuje tabuľka 4 – 2.

Počet disperzií	Početnosť udalostí
1	714
2	906
3	1029
4	337
5	79
6	27
7	16
8	2
9	4

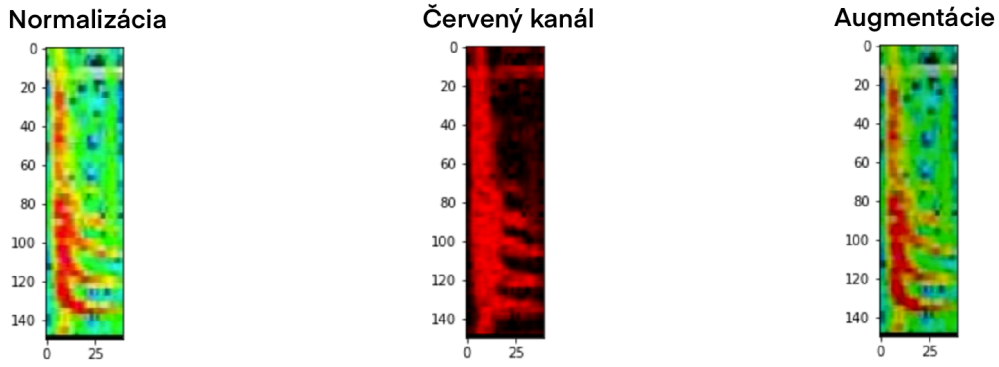
Tabuľka 4 – 1 Početnosť udalostí

Počet disperzií	Početnosť udalostí
1	714
2	906
3	1029
4 a viac	465

Tabuľka 4 – 2 Početnosť udalostí po spojení tried

Týmto krokom sa nám zároveň zredukoval počet klasifikačných tried na 4. Posledným krokom pred extrakciou bolo rozdelenie udalostí podľa počtu disperzií do samostatných csv súborov. Pomocou takto pripravených datasetov sme následne extrahovali jednotlivé udalosti zo spektrogramov a uložili ich do zložiek podľa počtu disperzií. Extrakcia prebiehala na základe koordinátov *xmin, ymin, xmax, ymax* a prislúchajúci spektrogram obsahujúci udalosť na extrakciu bol nájdený porovnaním mena spektrogramu a mena udalosti. Výsledkom tohto kroku predspracovania boli extrahované udalosti vo formáte jpg uložené v prislúchajúcich zložkách. Extrahované udalosti sme ešte rozdelili pomocou Python knižnice `splitfolders` na trénovaciu

a testovaciu zložku v pomere 90% trénovanie a 10% testovanie. Rozdelenie udalostí bolo realizované náhodným výberom. Vo fáze trénovania modelu sa z trénovacej zložky vyčlenila časť dát pre validáciu modelu. Validačná zložka predstavovala 20% z trénovacej zložky.



Obrázok 4 – 1 Prístupy predspracovania dát

V nasledujúcom kroku sme sa už zamerali na predspracovanie extrahovaných jpg súborov do podoby vhodnej pre trénovanie modelu. Z dôvodu malých rozdielov medzi jednotlivými triedami príkladov sme pre čo najlepšie učenie modelu vyskúšali tri rôzne prístupy predspracovania dát zobrazené na Obr. 4 – 1. Vo všetkých troch prístupoch sme využívali Python knižnicu `open-cv` na extrakciu informácií z obrázkov do číselnej reprezentácie pochopiteľnej pre model. Nasledujúce podkapitoly popisujú tri rôzne spôsoby predpracovania dát, na ktorých bol natrénovaný model klasifikácie disperzií.

Normalizácia

Pri predspracovaní spôsobom normalizácie sme obrázky normalizovali na rovnakú veľkosť 150×40 pixelov. Túto veľkosť sme určili tak, aby vynikli disperzie tweekov a model sa ich mohol lepšie naučiť rozlišovať. Po normalizácii sme číselné reprezentácie obrázkov uložili do premennej typu `array` vo formáte číselná reprezentácia obrázka s prislúchajúcou triedou, do ktorej patrí. Takto predspracované dáta sme

následne premiešali, aby sa jednotlivé triedy nevyskytovali bezprostredne za sebou a rozdelili ich na x a y zložky, kde x sú obrázky, na ktorých bude trénovaný model a y sú triedy, do ktorých obrázky prislúchajú. Rovnakým spôsobom sme predspracovali trénovaciu aj testovaciu množinu. Nakoniec sme za použitia knižnice `pickle` uložili predspracované dáta, aby sme proces predspracovania už nemuseli opakovať pri novom trénovaní modelu. Trénovacia zložka pozostávala z **2322** príkladov a testovacia zložka obsahovala **175** príkladov. Oproti početnostiam jednotlivých tried v tabuľke 4 – 2 je udalostí menej zhruba o 600. Je to z toho dôvodu, že sme v našej práci nemali dostupné všetky spektrogrami, ktoré boli anotované počas anotačného projektu a preto je extrahovaných obrázkov menej.

Červený kanál

Predspracovanie formou vytiahnutia informácie len červeného kanálu obrázkov prebiehalo rovnakým spôsobom ako predspracovanie normalizáciou. Rozdiel bol však v tom, že pri spracovaní obrázku do číselnej reprezentácie sme načítali hodnotu iba pre červený kanál. Túto operáciu sme robili za účelom lepšieho vyniknutia charakteristík jednotlivých tried. Predspracované dáta sme opäť uložili pomocou knižnice `pickle` a aj v tomto prípade pozostávala trénovacia zložka z **2322** príkladov a testovacia zo **175** príkladov.

Augmentácie

Pri augmentácii dát sme používali dáta predspracované spôsobom normalizácie, ktoré sme načítali z `pickle` súboru. Na realizáciu augmentácii sme použili knižnice `Albumentations` a `ImageDataAugmentor`. Druhá menovaná knižnica je modifikáciou knižnice `ImageDataGenerator`, ktorá je upravená aby dokázala fungovať s knižnicou `Albumentations`. Na vstupných obrázkoch sme menili jas, kontrast a gammu v ohraničených hodnotách a s prednastavenou pravdepodobnosťou. Nastavenie augmentoru je popísané tabuľkou 4 – 3. Rozsah hodnôt sme nastavili tak, aby augmentované obrázky zodpovedali udalostiam, ktoré sa môžu reálne vyskytnúť na

spektrogramoch. Z toho dôvodu sme nevyužívali augmentačné operácie otáčania a zmeny veľkosti, pretože na spektrogramoch sa také udalosti nevyskytujú.

Operácia	min hodnota	max hodnota	pravdepodobnosť
`RandomGamma`	100	200	50%
`RandomBrightness`	-0,05	0,15	50%
`RandomContrast`	-0,05	0,1	50%

Tabuľka 4 – 3 Nastavenie augmentoru

Po nastavení augmentoru sme inicializovali `data generator`, v ktorom sme nastavili `seed`[1], priradili mu vytvorený augmentor a nastavili validačnú množinu, ktorá tvorí 20% z trénovacej množiny. Pri trénovaní modelu sa tento generátor spustil a na každom *batchy* dát počas trénovania generoval nové augmentované dáta s prednastavenými parametrami a pravdepodobnosťou viď. tabuľka 4 – 3. Výhodou augmentácií je rozšírenie datasetu o nové a pozmenené príklady, na ktorých sa model môže lepšie naučiť rozpoznávať jednotlivé typy udalostí.

4.2 Modelovanie

Vytvoreniu konečného modelu predchádzalo rozsiahle testovanie rôznych kombinácii parametrov modelu, veľkosti modelu ale aj spôsobu trénovania. Popísaný model predstavuje najlepšie nájdené riešenie z hľadiska výkonnosti a rýchlosti modelu.

Konštrukcia modelu

Vytvorený model konvolučnej neurónovej siete sa skladá zo štyroch konvolučných vrstiev `Conv2D` s veľkosťou filtra 3×3. U všetkých štyroch vrstiev sme použili aktivačnú funkciu ReLu a počet konvolučných filtrov je 16 a 32 v prípade prvých dvoch vrstiev (model naučený na augmentovanej množine má počet prvých dvoch filtrov 32 a 64), a počet filtrov zostávajúcich vrstiev je 64. Každá konvolučná vrstva je

[1] `seed` používame pre reprodukovateľnosť výsledkov

nasledovaná vrstvou `MaxPooling2D` s veľkosťou okna 2×2 a nastaveným `paddingom` na hodnotu `same`. To zabezpečí, že aj obrázok veľkosti 150 x 40 pixelov bude možné spracovať sieťou. Výstup posledného `MaxPooling2D` je nasledovaný `Flatten` vrstvou. Aby sme zabránili preučeniu CNN používame za touto vrstvou `Dropout` nastavený na hodnotu 0.2 (model naučený na augmentovanej množine má hodnotu dropoutu 0.3). Poslednými dvoma vrstvami sú `Dense` vrstvy, kde prvá má počet neurónov 128 a aktivačnú funkciu ReLu. Keďže klasifikujeme do štyroch tried, má výstupná `Dense` vrstva počet neurónov 4, každý reprezentujúci jednu triedu a použitá aktivačná funkcia je Softmax.

Pri kompilácii modelu sme použili optimalizačnú funkciu `adam`, chybovú funkciu `sparse_categorical_crossentropy` a sledovanú metriku `accuracy`. Parameter `checkpoint` sme nastavili tak, aby sa pri trénovaní ukladal iba najlepší model z hľadiska monitorovanej `val_loss` validačnej chyby. Posledný parameter, ktorý sme nastavovali, bol parameter `early stopping` a nastavili sme ho tak, aby predčasne ukončil trénovanie modelu, ak sa nezlepší presnosť na validačnej množine počas piatich za sebou idúcich epochách.

Trénovanie modelu

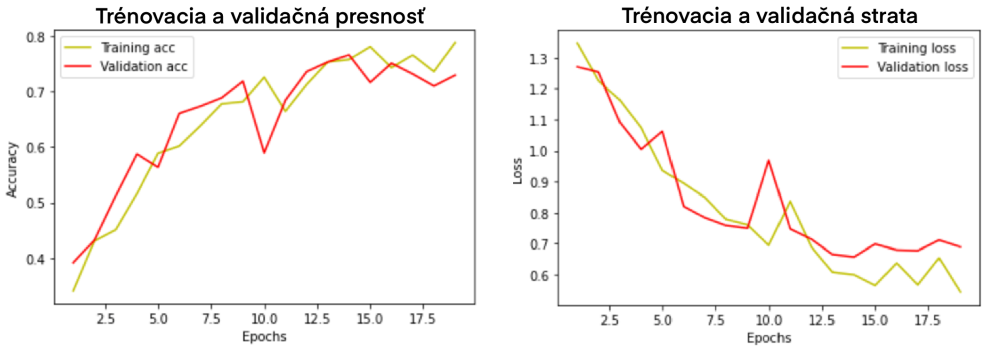
Trénovanie modelov na predspracovaných dátach normalizáciou a vytiahnutím červeného kanála prebiehalo rovnakým spôsobom. Trénovanie sa spustilo príkazom:

```
history = model.fit(X, y, epochs = 30, validation_split = 0.2,
batch_size = 64, callbacks = callbacks_list)
```

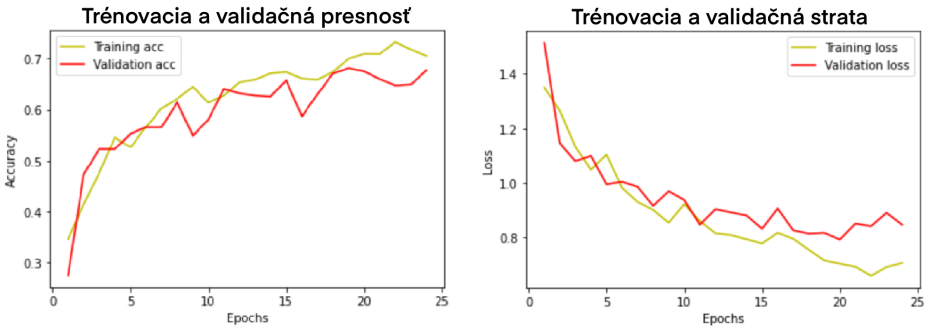
- `X` – trénovacie dáta
- `y` – label trénovacích dát
- `epochs` – počet epoch trénovania
- `validation_split` – rozdelenie trénovacích dát na validačnú podmnožinu v pomere 80% trénovanie a 20% validácia

- `batch_size` – veľkosť dávky dát
- `callbacks` – zoznam preddefinovaných kritérií (v našom prípade `checkpoint` a `early`)

Trénovanie CNN sa v oboch prípadoch skončilo predčasne, naplnením kritéria `early stopping`. Model 1 trénovaný na dátach predspracovaných normalizáciou ukončil trénovanie po 19 epochách a Model 2 trénovaný na dátach s červeným kanálom po 24 epochách. Priebehy trénovania oboch modelov sú znázornené na obrázkoch 4 – 2 a 4 – 3.



Obrázok 4 – 2 Priebeh učenia – Model 1



Obrázok 4 – 3 Priebeh učenia – Model 2

Spustenie trénovania modelu s augmentovanými dátami na vstupe prebiehalo trochu odlišne. Najprv sme inicializovali `datagen`:

```
datagen = ImageDataAugmentor( seed = SEED, augment=AUGMENTATIONS,
validation_split=0.2)
```

- `seed` – prednastavená hodnota pre reprodukovateľnosť výsledkov
- `augment` – nastavenie augmentácií (4 – 3)
- `validation_split` – rozdelenie na validačnú podmnožinu v pomere 80% trénovanie a 20% validácia

Následne sa na inicializovanom generátore spustili trénovacie dáta:

```
datagen.fit(x, augment=True)
```

- `x` – trénovacie dáta
- `augment` – parameter určujúci, či sa budú dáta augmentovať

```
train_generator = datagen.flow
(x, y, batch_size=32, subset='training', shuffle = False)
validation_generator = datagen.flow
(x, y, batch_size=32, subset='validation', shuffle = False)
```

- `x` – trénovacie dáta
- `y` – label trénovacích dát
- `batch_size` – veľkosť dávky dát
- `subset` – označenie podmnožiny dát
- `shuffle` – premiešanie poradia dát

Nakoniec sme spustili trénovanie modelu príkazom:

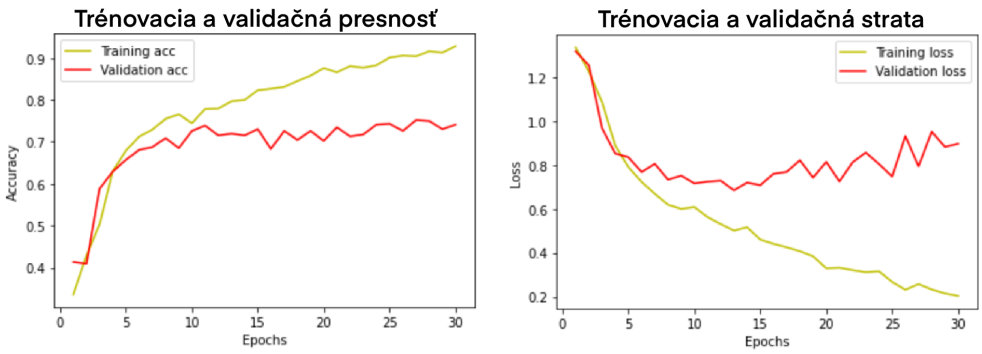
```
history = model.fit(
   train_generator,
```

```
    steps_per_epoch=len(train_generator),
    epochs=30,
    validation_data=validation_generator,
    validation_steps=len(validation_generator),
    callbacks = callbacks_list
```

), kde

- `train_generator` – generátor inicializovaný na trénovacích dátach
- `steps_per_epoch` – počet krokov na epochu
- `epochs` – počet epoch trénovania
- `validation_data` – validačná dátová množina
- `validation_steps` – počet validačných krokov
- `callbacks` – zoznam preddefinovaných kritérii (`checkpoint`)

Keďže Model 3 neobsahoval ukončovacie kritérium `early stopping`, tak zbehlo všetkých 30 epoch trénovania. Priebeh trénovania modelu je znázornený na Obr. 4 – 4. Pri tŕenovaní sa ukladal najlepší model z hľadiska validačnej straty.



Obrázok 4 – 4 Priebeh učenia – Model 3

4.3 Vyhodnotenie

Celkovo sme na klasifikáciu disperzií tweekov natrénovali tri modely. Vytvorené modely sme vyhodnocovali dvoma spôsobmi. Prvým spôsobom bolo vyhodnotenie z hľadiska presnosti a návratnosti modelov. Najlepšie natrénovaný model sme spustili na testovacej množine príkladov skladajúcej sa zo **175** udalostí a porovnali sme klasifikáciu modelu, so skutočnou triedou príkladov a z porovnania sme vytvorili kontingenčnú tabuľku. Kontingenčná tabuľka je reprezentovaná tabuľkou 4 – 4.

True Positive (TP)	False Positive (FP)
False Negative (NP)	True Negative (TN)

Tabuľka 4 – 4 Kontingenčná tabuľka

Z kontingenčnej tabuľky sme následne vypočítali presnosť (*angl. precision*) jednotlivých tried, návratnosť (*angl. recall*), f1 skóre (*angl. f1-score*) a úspešnosť (*angl. accuracy*) modelu pomocou vzorcov4.1, 4.2, 4.3, 4.4:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN} \tag{4.3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

Druhým spôsobom bolo vyhodnotenie vizuálnou kontrolou nesprávne zaklasifikovaných príkladov. Nesprávne zaklasifikované príklady sme vizualizovali spolu s údajom o klasifikácii modelu a skutočnou triedou. Keďže rozdiely medzi jednotlivými triedami sú v niektorých prípadoch veľmi malé a ani doménový expert niekedy

nedokáže určiť jednoznačne príslušnosť príkladu do konkrétnej triedy, chceli sme klasifikácie modelu ešte dodatočne overiť. V následujúcich podkapitolách sú popísané vyhodnotenia vytvorených modelov.

Model 1

Model bol natrénovaný na dátach predspracovaných normalizáciou a vyhodnotený na testovacej množine pozostávajúcej zo **175** príkladov. Kontingenčná tabuľka klasifikácie je znázornená tabuľkou 4 – 5 a výkonnosť modelu tabuľkou 4 – 6.

	Tweek1	Tweek2	Tweek3	Tweek4+
Tweek1	43	2	1	0
Tweek2	1	21	1	2
Tweek3	1	5	55	1
Tweek4+	0	0	9	33

Tabuľka 4 – 5 Kontingenčná tabuľka – Model 1

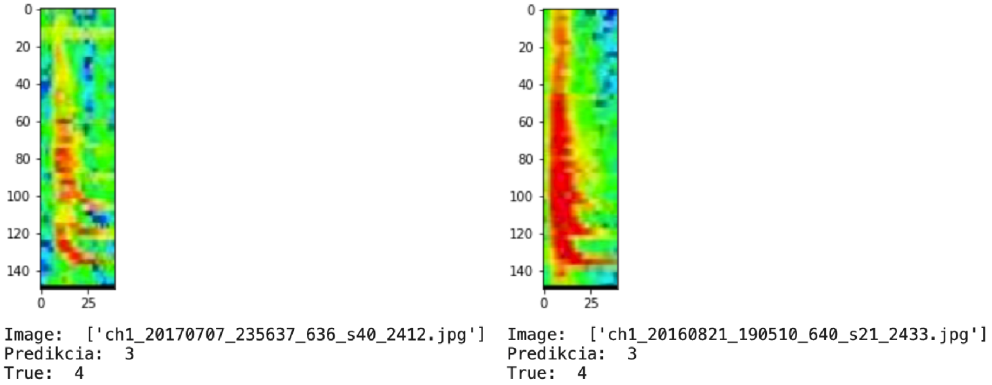
Trieda	Presnosť	Návratnosť	F1 skóre	Počet
Tweek1	0,96	0,93	0,95	46
Tweek2	0,75	0,84	0,79	25
Tweek3	0,83	0,89	0,86	62
Tweek4+	0,92	0,79	0,86	42
Úspešnosť			0,87	175
Makro priemer	0,86	0,86	0,86	175
Vážený priemer	0,87	0,87	0,87	175

Tabuľka 4 – 6 Výkonnosť – Model 1

Z tabuľky 4 – 6 vidíme, že model dosiahol úspešnosť 86% pri makro priemere a 87% pri váženom priemere. Rozdiel medzi týmito dvoma metrikami je v tom, že makro priemer neberie do úvahy proporcionálne rozloženie tried v datasete. Z

presnosti jednotlivých tried vidíme, že najlepšie model klasifikoval triedy `Tweek1` (96%) a `Tweek4+` (92%). Triedy `Tweek2` a `Tweek3` model klasifikoval s presnosťou 75% a 83%.

Z vizuálneho overenia zobrazenom na Obr. 4 – 5 sme zistili, že niektoré obrázky by mohli patriť aj do triedy predikovavej modelom, nakoľko ani my sami sme sa nevedeli rozhodnúť, či by sme vizualizované príklady zaradili do triedy `Tweek3` alebo `Tweek4+`. Výsledná presnosť modelu teda môže byť ešte vyššia.



Obrázok 4 – 5 Vizualizácia predikcií – Model 1

Model 2

Model bol natrénovaný na dátach obsahujúcich iba hodnoty červeného kanála a vyhodnotený na testovacej vzorke pozostávajúcej zo **175** príkladov. Tabuľka 4 – 7 predstavuje kontingenčnú tabuľku modelu a tabuľka 4 – 8 výkonnosť modelu.

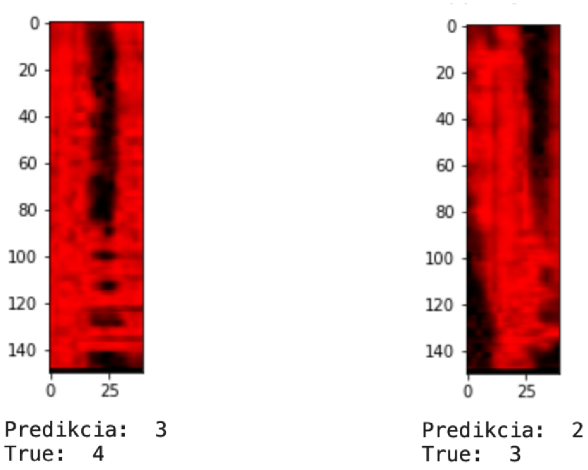
	`Tweek1`	`Tweek2`	`Tweek3`	`Tweek4+`
`Tweek1`	39	4	3	0
`Tweek2`	0	21	2	2
`Tweek3`	0	6	52	4
`Tweek4+`	1	1	12	28

Tabuľka 4 – 7 Kontingenčná tabuľka - Model 2

Trieda	Presnosť	Návratnosť	F1 skóre	Počet
`Tweek1`	0,97	0,85	0,91	46
`Tweek2`	0,66	0,84	0,74	25
`Tweek3`	0,75	0,84	0,79	62
`Tweek4+`	0,82	0,67	0,74	42
Úspešnosť			0,80	175
Makro priemer	0,80	0,80	0,79	175
Vážený priemer	0,81	0,80	0,80	175

Tabuľka 4 – 8 Výkonnosť – Model 2

Z tabuľky 4 – 7 vidíme zhoršenú úspešnosť klasifikácie oproti predchádzajúcemu modelu. Úspešnosť dosahuje 80% makro respektíve 81% vážený priemer. Pri predikcii jednotlivých tried mal model najväčší problém s triedami `Tweek2` a `Tweek3`. Naopak triedu `Tweek1` vedel predikovať dobre. Vo vizuálnom overení (Obr. 4 – 6) je vidieť značné skreslenie udalostí. Tým, že obrázky obsahujú len hodnoty pre červený kanál vznikol problém skresľovania udalostí, ak boli na spektrograme zachytené veľmi blízko seba. Tento fakt sťažuje predikovanie modelu, čo sa odráža na zhoršenej výkonnosti.



Obrázok 4 – 6 Vizualizácia predikcií – Model 2

Model 3

Posledným vyhodnoteným modelom, bol model trénovaný na augmentovanej dátovej množine. Vďaka rozšíreniu dátovej množiny o nové augmentované príklady sa dokázal model naučiť rozpoznávať jednotlivé druhy tried najlepšie. Už z kontingenčnej tabuľky zobrazenej na tabuľke 4 – 9 vidíme malé množstvo nesprávne zaklasifikovaných príkladov a tabuľka 4 – 10 ďalej potvrdzuje dobrú výkonnosť modelu, s úspešnosťou klasifikácie na úrovni 92%. Model sa dokázal vysporiadať aj s menej početnou triedou `Tweek2`, ktorú predikuje s presnosťou 85%. Hodnoty presnosti ostatných tried sú taktiež na veľmi dobrej úrovni, rovnako ako aj hodnoty návratnosti a f1 skóre.

	`Tweek1`	`Tweek2`	`Tweek3`	`Tweek4+`
`Tweek1`	44	1	1	0
`Tweek2`	1	22	2	0
`Tweek3`	1	3	57	1
`Tweek4+`	0	0	4	38

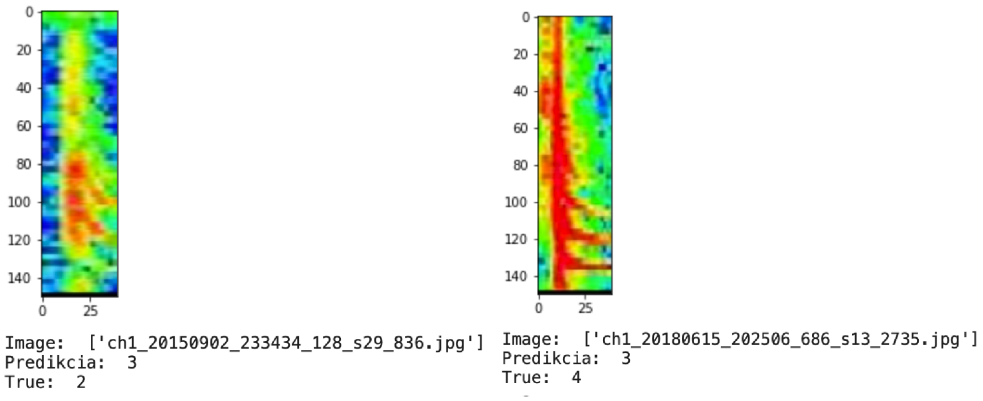
Tabuľka 4 – 9 Kontingenčná tabuľka – Model 3

Trieda	Presnosť	Návratnosť	F1 skóre	Počet
`Tweek1`	0,96	0,96	0,96	46
`Tweek2`	0,85	0,88	0,86	25
`Tweek3`	0,89	0,92	0,90	62
`Tweek4+`	0,97	0,90	0,94	42
Úspešnosť			0,92	175
Makro priemer	0,92	0,92	0,92	175
Vážený priemer	0,92	0,92	0,92	175

Tabuľka 4 – 10 Výkonnosť – Model 3

Vizuálnym overením (Obr. 4 – 7) sa nám opäť potvrdila spornosť priradenia niekto-

rých príkladov. Dané ukážky by totiž kľudne mohli patriť aj do triedy predikovanej modelom. Výsledná presnosť modelu teda opäť môže byť vyššia ako je tabuľkové vyhodnotenie. Ako je zrejmé z vizuálnych vyhodnotení, anotácie udalostí v anotačnom projekte nemusia byť presné na 100% a mohlo dôjsť k chybám pri anotovaní, nakoľko vizuálne odhadnúť príslušnosť zachytenej udalosti s počtom jej disperzií je v niektorých prípadoch veľmi ťažké. Tieto vizualizované chyby potvrdzujú, že model dokáže počet disperzií predikovať lepšie, ako niektorí naši anotátori. Keďže tento model má najlepšiu výkonnosť zo všetkých troch vytvorených modelov, použijeme ho ďalej vo fáze nasadenia na klasifikáciu nameraných dát prístrojom ELMAVAN-G.



Obrázok 4 – 7 Vizualizácia predikcií – Model 3

4.4 Nasadenie

V tejto kapitole popíšeme celý postup nasadenia a rozšírenia predchádzajúceho výskumu od stiahnutia dát, až po výstupné tabuľky a ich analýzu. Pod nasadením rozumieme spustenie vytvorených metód na nameraných dátach. Vytvorenými metódami sme spracovali namerané dáta z roku 2016 a pre potreby nekontrolovaného učenia aj menšiu vzorku dát z roku 2014. Postup môžeme rozdeliť na nasledujúce podúlohy:

1. **Stiahnutie dát**

Dáta namerané prístrojom ELMAVAN-G sa nachádzajú na vzdialenom úložisku a pre ďalšiu prácu s nimi je potrebné ich stiahnuť na lokálne úložisko. Stiahnutie všetkých dát má veľkosť vyše 5 terabajtov a preto nám bol poskytnutý dátovy server Technickej univerzity v Košiciach, ktorý tieto pamäťové požiadavky dokázal pokryť. Dáta sme zo vzdialeného úložiska sťahovali konzolovým príkazom `wget`.

2. **Rozbalenie dát**

   Dáta boli stiahnuté vo formáte zip a pred ďalším spracovaním ich bolo potrebné rozbaliť. Pre minimalizovanie času spracovania sme dáta rozbaľovali paralelne po mesiacoch. To znamená, že v jednom okamihu sa rozbaľovali súčasne všetky mesiace jedného roku.

3. **Spracovanie pdf**

   Po rozbalení sme dostali pdf súbor, ktorý obsahoval 9 spektrogramov, každý dĺžky jednej sekundy, počas ktorej bolo meranie najsilnejšie z celkového 144 sekundového merania. Tieto spektrogramy bolo potrebné extrahovať a skonvertovať do formátu jpg, ktorý je vhodný pre YOLO detekciu. Tento proces bol časovo a výpočtovo najnáročnejší. Pri paralelnom spracovaní každého mesiaca nameraných dát súčasne zabral tento krok vyše dvoch dní čistého času spracovania.

4. **YOLO detekcia**

   Na stiahnutých a predspracovaných dátach bolo následne možné spustiť natrénovaný YOLO model na detekciu udalostí. Pri detekcii YOLO označuje udalosti na spektrogramoch rámčekmi, ktoré sú farebne odlíšené podľa klasifikačných tried. YOLO však má možnosť nájdené udalosti a informácie o nich ukladať aj vo forme txt súborov. Tieto súbory obsahujú informáciu o druhu detegovanej udalosti a súradnice rámčeka, ktorým bol označený. Na základe informácií obsiahnutých v txt súboroch ďalej extrahujeme jednotlivé

typy udalostí. Tento a všetky predchádzajúce popísané kroky boli súčasťou nášho predchádzajúceho výskumu (Maslej-Krešňáková et al., 2021). Nasledujúce kroky vytvorenia výstupných tabuliek boli modifikované a rozšírené o nové vytvorené metódy.

5. **Extrakcia udalostí typu sferik a vytvorenie výstupnej tabuľky sferikov**

   Podľa informácií z txt súborov extrahujeme zo spektrogramov udalosti typu sferik a uložíme ich do samostatnej zložky ako jpg súbory. Následne vytvoríme čiastkovú výstupnú tabuľku sferikov pre neskoršie spojenie. Pre vytvorenie konečnej výstupnej tabuľky nepotrebujeme sferiky extrahovať a ukladať do samostatnej zložky. Tento krok však robíme pre potreby metód nekontrolovaného učenia.

6. **Extrakcia udalostí typu tweek a vytvorenie výstupnej tabuľky tweekov**

   Tento krok je v podstate totožný s predchádzajúcim krokom, ale extrahujeme udalosti typu tweek. Narozdiel od predchádzajúceho kroku je však extrakcia udalostí potrebná pre spracovanie dát do podoby vhodnej pre klasifikáciu modelom na určovanie počtu disperzií. V tomto kroku sa taktiež vytvorí čiastková výstupná tabuľka tweekov.

7. **Klasifikácia tweekov podľa počtu disperzií a rozšírenie výstupnej tabuľky tweekov o tento údaj**

   Extrahované tweeky sa spracujú do podoby vhodnej pre predikciu modelom. Načíta sa natrénovaný model a následne sa spustí predikcia na dátach. Model zaklasifikuje každý tweek do triedy podľa počtu disperzií. Následne sa rozšíri čiastková výstupná tabuľka tweekov o nový stĺpec obsahujúci informáciu o počte disperzií.

8. **Spojenie výstupnej tabuľky sferikov a tweekov do konečnej výstupnej tabuľky**

   Nakoniec nám ostáva pre vytvorenie konečnej výstupnej tabuľky už len spojiť čiastkovú výstupnú tabuľku sferikov a tweekov. Po spojení udalosti v tabuľke usporiadame aby išli za sebou tak, ako boli zachytené YOLO algoritmom.

9. **Vymazanie duplicitných detekcií**

   Úplne posledným krokom je vymazanie duplicitných záznamov z výstupnej tabuľky. Jedným z problémov YOLO algoritmu je, že niekedy označí rovnakú udalosť oboma triedami. Dôvodom duplicitného označenia tried je nízko nastavená citlivosť detektora udalostí. V našom prípade označí model tú istú udalosť ako sferik a aj ako tweek. To by mohlo predstavovať problém pri ďalšej analýze a hlavne pri nekontrolovanom učení. Preto vytvorenou metódou duplicitné záznamy z tabuľky vymažeme. Mazanie prebieha na základe rovnakej milisekundy dvoch po sebe idúcich udalostí, ak je jedna udalosť typu tweek a druhá typu sferik. Z tabuľky mažeme udalosť typu sferik. V závislosti od nameraných dát sa duplicitná detekcia pohybovala na úrovni do $2,5\%$.

Výsledná výstupná tabuľka za rok 2016 obsahovala viac ako 4,4 milióna udalostí, z ktorých bolo okolo 200000 udalostí typu tweek. Tabuľka sa skladala zo stĺpcov:

- **org_img** – názov spektrogramu, z ktorého pochádza udalosť
- **image** – názov udalosti
- **event** – poradové číslo lokalizovanej udalosti na spektrograme (sferiky a tweeky majú svoje vlastné poradie)
- **date** – dátum a čas výskytu udalosti v tvare YYYYMMDD-HHMMSS
- **second** – sekunda zachytenia udalosti
- **milisecond** – presná milisekunda zachytenia udalosti

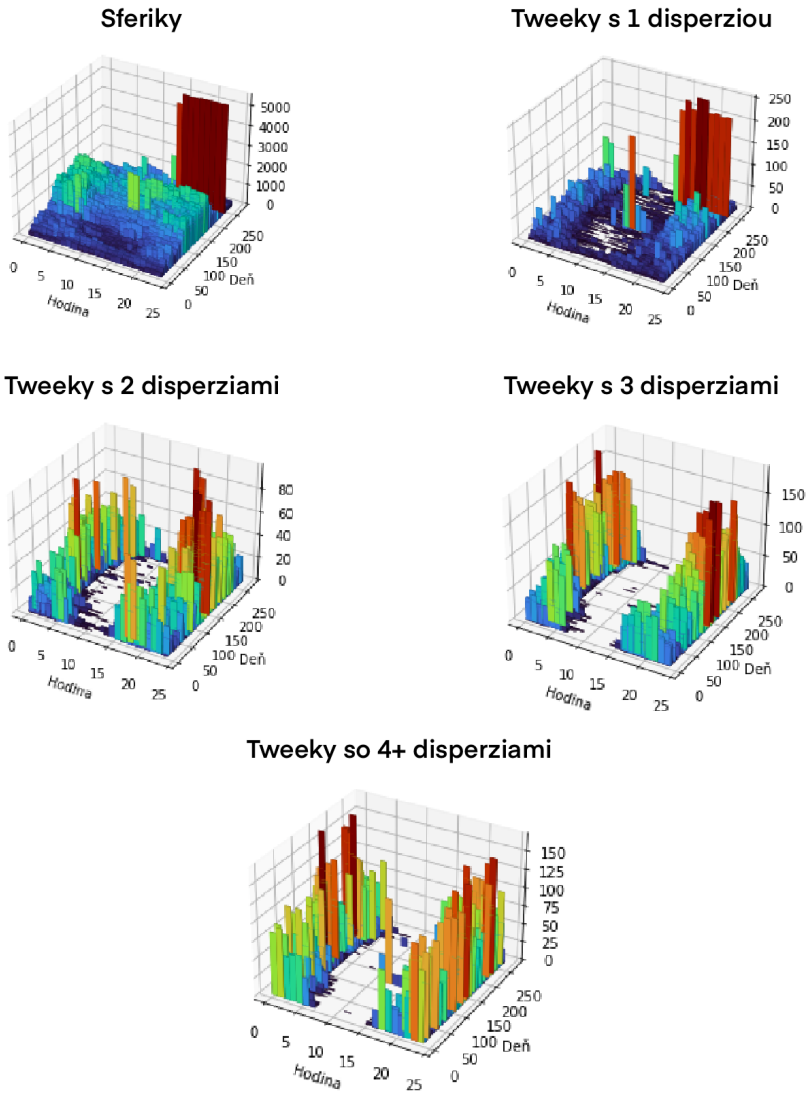
- **tweek** – binárna hodnota, kde 0 predstavuje sferik a 1 predstavuje tweek
- **f_min<2kHz** – binárna hodnota, kde 0 je udalosť, ktorej emisia je nad hranicou $2kHz$ a 1 je emisia pod hranicou $2kHz$
- **tweek_sign** – počet disperzií zachytenej udalosti

	org_img	image	event	date	second	milisecond	tweek	f_min<2kHz	tweek_sign
0	ch2_20160101_000242_596_s102	ch2_20160101_000242_596_s102-S1.jpg	1	20160101_000242	102	455	0	0	0
1	ch2_20160101_000242_596_s12	ch2_20160101_000242_596_s12-S1.jpg	1	20160101_000242	12	854	0	0	0
2	ch2_20160101_000242_596_s143	ch2_20160101_000242_596_s143-S1.jpg	1	20160101_000242	143	227	0	0	0
3	ch2_20160101_000242_596_s143	ch2_20160101_000242_596_s143-S2.jpg	2	20160101_000242	143	973	0	0	0
4	ch2_20160101_000242_596_s21	ch2_20160101_000242_596_s21-S1.jpg	1	20160101_000242	21	33	0	1	0
...	...	...	...	...	...	...	...	...	...
4417691	ch2_20161231_235851_604_s42	ch2_20161231_235851_604_s42-S1.jpg	1	20161231_235851	42	878	0	0	0
4417692	ch2_20161231_235851_604_s52	ch2_20161231_235851_604_s52-S1.jpg	1	20161231_235851	52	739	0	0	0
4417693	ch2_20161231_235851_604_s69	ch2_20161231_235851_604_s69-S1.jpg	1	20161231_235851	69	246	0	0	0
4417694	ch2_20161231_235851_604_s69	ch2_20161231_235851_604_s69-T1.jpg	1	20161231_235851	69	248	1	0	2
4417695	ch2_20161231_235851_604_s82	ch2_20161231_235851_604_s82-T1.jpg	1	20161231_235851	82	810	1	0	3

Obrázok 4 – 8 Výstupná tabuľka pre rok 2016

Na Obr. 4 – 8 je znázornený náhľad výstupnej tabuľky pre rok 2016. Tabuľka obsahuje všetky dôležité informácie o zachytených udalostiach. Ďalšou analýzou tejto tabuľky vieme získať zaujímave informácie, ktoré by mohli viesť k novým objavom. Veľmi zaujímavou informáciou pre nás bolo rozloženie početnosti jednotlivých typov udalostí počas dňa. Vieme totiž, že tweeky sa skoro vôbec nevyskytujú počas dňa. Zachytenie denných tweekov a ich skúmanie by mohlo viesť k úplne novým objavom. Pomocou vizualizačnej Python knižnice `Matplotlib` sme vytvorili 3D stĺpcové grafy frekvencie jednotlivých typov udalostí. Grafy sú znázornené na Obr. 4 – 9. Na osi x je znázornený čas v hodinách, os y znázorňuje dni v roku a na osi z je znázornená početnosť zachytených udalostí. Grafy tweekov s dvoma a viac disperziami potvrdzujú trend majoritného výskytu počas noci. V grafoch sa však vyskytujú aj ojedinelé nárasty počas dňa. Taktiež môžeme pozorovať zvýšený výskyt sferikov a

tweekov s jednou disperziou ku koncu roka. Prečo nastali tieto javy je predmetom aktuálneho výskumu v spolupráci s Ústavom fyziky atmosféry Akademie věd České republiky. Vývoj vizualizovaných distribúcii počas jednotlivých rokov by taktiež mohol predstavovať zaujímavý predmet budúceho výskumu.



Obrázok 4 – 9 Grafy frekvenčného výskytu zachytených udalostí

5 Nekontrolované učenie

Nekontrolované učenie sme realizovali formou metód k-Means v kombinácií s autoenkóderom a taktiež sme vyskúšali aj metódu samo-organizujúcich sa máp. Cieľom bolo nájsť v dátach nové vzory a zaujímavé súvislosti, ktoré by metódami kontrolovaného učenia nebolo možné objaviť. V nasledujúcich podkapitolách popíšeme proces prípravy dát pre metódy nekontrolovaného učenia a taktiež popíšeme modelovanie a obmeny jednotlivých vyskúšaných metód. Nakoniec interpretujeme výsledky dosiahnuté nekontrolovaným učením.

5.1 Príprava dát

Príprava dát bola v prípade metód nekontrolovaného učenia jednoduchá, nakoľko väčšinu krokov predspracovania sme už urobili pri vytváraní klasifikačného modelu disperzií tweekov. Jedná sa o kroky extrakcie jednotlivých udalostí a ich uloženie do samostatných zložiek. Využívali sme dáta, ktoré boli extrahované v procese nasadenia modelu klasifikácie disperzií. Tento proces je popísaný v podkapitole 4.4. Extrahované udalosti sme pomocou Python knižnice `open-cv` transformovali do číselnej reprezentácie, normalizovali ich na veľkosť 150 x 40 pixelov a uložili do premennej typu `array`. Následne sme premiešali poradie uložených dát pomocou príkazu `shuffle` a vytvorili z nich `pickle` súbor. Takto predspracované dáta sme použili v oboch realizovaných metódach. Použitá vzorka dát pochádzala z roku 2014 a obsahovala **13647** udalostí. Autoenkóder bol naučený na dátovej množine, ktorá bola predspracovaná spôsobom popísaným v podkapitole 4.1. Tieto predspracované dáta sa skladajú z **2322** udalostí a načítali sme ich z `pickle` súboru.

5.2 Autoenkóder + k-Means

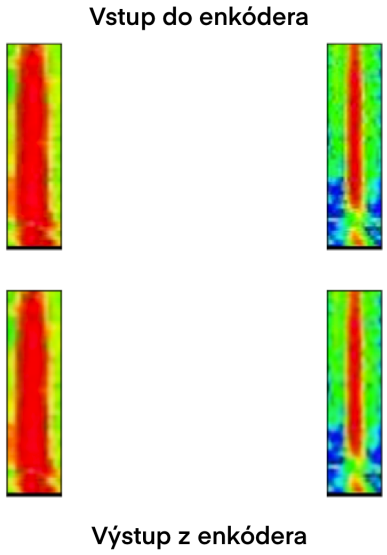
Metódu k-Means sme vybrali z dôvodu jej jednoduchosti a výpočtovej nenáročnosti. Vyskúšali sme ju samostatne ale aj v kombinácii s autoenkóderom. Natrénovali sme dva modely autoenkóderu, z ktorých prvý redukuje dimenzionalitu dát zo $150 \times 40 \times 3$

na $25 \times 10 \times 64$ a druhý na $25 \times 10 \times 32$. Oba enkódery sa prakticky líšia len počtom konvolučných filtrov. Vzhľadom na menšiu veľkosť dátovej množiny sme sa rozhodli pre použitie autoenkódera, ktorý redukuje dimenzinalitu na $25 \times 10 \times 64$. Na zakódovanej reprezentácii dát sme následne spustili zhlukovanie k-Means.

5.2.1 Modelovanie

Vytvorený autoenkóder pozostával z enkódovacej a dekódovacej časti. Na začiatku enkódovacej časti sme ako prvé definovali veľkosť vstupu na $150 \times 40 \times 3$. Enkóder sa následne skladal z dvoch konvolučných vrstiev `Conv2D` s veľkosťou filtra 3×3. Prvá vrstva obsahovala 128 konvolučných filtrov a v druhej sme počet zmenšili na 64 filtrov. Použitou aktivačnou funkciou bola ReLu. Za každou konvolučnou vrstvou nasledovala vrstva `MaxPooling2D`. Prvá `MaxPooling2D` vrstva mala veľkosť okna 3×2 a druhá 2×2. Veľkosti okna boli zvolené vzhľadom na veľkosť vstupu, aby ho bolo možné v časti dekóderu zrekonštruovať do pôvodnej veľkosti. Časť dekóderu pozostáva opäť z dvoch konvolučných vrstiev `Conv2D` s veľkosťou filtrov 3×3 a aktivačnou funkciou ReLu. Prvá vrstva má počet konvolučných filtrov 64 a druhá 128. Obe vrstvy nasleduje vrstva `UpSampling2D` s veľkosťou okna 2×2 a 3×2. Poslednou vrstvou dekódera je konvolučná vrstva `Conv2D` s filtrom veľkosti 3×3 a troma konvolučnými filtrami. V tomto prípade je použitá aktivačná funkcia sigmoid. Použitá bola optimalizačná funkcia `adam` a chybová funkcia `binary_crossentropy`.

Fungovanie autoenkóderu na vstupných dátach znázorňuje Obr. 5 – 1, kde vidíme, že rekonštruované vstupy sú stále dobre čitateľné. Zakódovaná reprezentácia dát, ktorú využívame ako vstup metódy k-Means je zobrazená na Obr. 5 – 2. Zakódovaná reprezentácia obsahuje dôležité črty dát, pri zredukovanej dimenzionalite pre rýchlejšie spracovanie. Takto naučeným autoenkóderom sme následne zakódovali naše vstupné dáta. Zakódovaná reprezentácia mala tvar (13647, 25, 10, 64). Tento tvar však nie je vhodný pre k-Means zhlukovanie a preto sme ho pomocou funkcie `reshape` zmenili na (13647, 16000). Na takto pripravených dátach sme spustili zhlukovaciu metódu k-Means.



Obrázok 5 – 1 Rekonštrukcia vstupov autoenkóderom

Obrázok 5 – 2 Zakódovaná reprezentácia vstupných dát piatich udalostí

Pomocou k-Means metódy zhlukovania sme dáta zhlukovali do dvoch až deviatich zhlukov. Vytvorené zhluky sme vizualizovali pomocou knižnice `matplotlib` tak, že sme vykreslili prvých 10 reprezentantov zhluku. Bol to dostatočný počet na vizuálne skúmanie odlišností zhlukov. Pre ďalšiu analýzu zhlukovaných dát sme výstup zhlukovacej metódy k-Means spojili s výstupnou tabuľkou vytvorenou v procese 4.4.

Náhľad výstupnej tabuľky predstavuje Obr. 5 – 3 a vidíme v nej doplnený stĺpec *cluster*, ktorý nám hovorí, do ktorého zhluku bola udalosť priradená. Pomocou vizuálnej kontroly zhlukov a informácií obsiahnutých vo výstupnej tabuľke vieme interpretovať vytvorené zhluky.

	org_img	image	event	date	second	milisecond	tweek	f_min<2kHz	tweek_sign	cluster
0	ch2_20141130_000500_988_s122	ch2_20141130_000500_988_s122-S1.jpg	1	20141130_000500	122	867	0	0	0	0
1	ch2_20141130_000500_988_s122	ch2_20141130_000500_988_s122-T1.jpg	1	20141130_000500	122	990	1	1	4	1
2	ch2_20141130_000500_988_s122	ch2_20141130_000500_988_s122-S2.jpg	2	20141130_000500	122	992	0	1	0	1
3	ch2_20141130_000500_988_s125	ch2_20141130_000500_988_s125-S1.jpg	1	20141130_000500	125	5	0	1	0	0
4	ch2_20141130_000500_988_s125	ch2_20141130_000500_988_s125-S2.jpg	2	20141130_000500	125	158	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
13642	ch2_20141130_235542_205_s9	ch2_20141130_235542_205_s9-S13.jpg	13	20141130_235542	9	824	0	1	0	0
13643	ch2_20141130_235542_205_s9	ch2_20141130_235542_205_s9-S14.jpg	14	20141130_235542	9	884	0	0	0	1
13644	ch2_20141130_235542_205_s9	ch2_20141130_235542_205_s9-S15.jpg	15	20141130_235542	9	916	0	0	0	0
13645	ch2_20141130_235542_205_s9	ch2_20141130_235542_205_s9-S16.jpg	16	20141130_235542	9	939	0	0	0	0
13646	ch2_20141130_235542_205_s95	ch2_20141130_235542_205_s95-S1.jpg	1	20141130_235542	95	975	0	1	0	1

Obrázok 5 – 3 Výstupná tabuľka s informáciou o zhlukoch

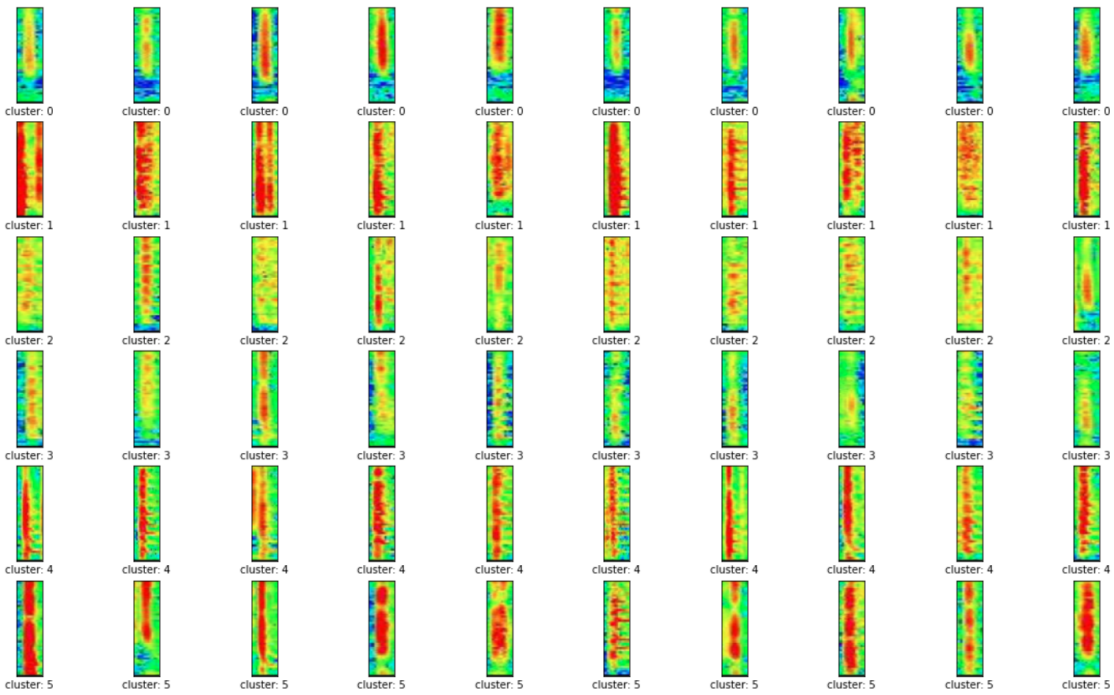
5.2.2 Interpretácia výsledkov

Analýzou vytvorených metód zhlukovania sme zistili, že najlepšie výsledky dosahuje zhlukovanie do šiestich zhlukov. Pri menšom počte zhlukov sa jednotlivé zhluky nedajú dobre odlíšiť a pri príliš veľkom počte zhlukov sú si jednotlivé zhluky veľmi podobné. V tejto podkapitole preto podrobne rozoberieme práve k-Means zhlukovanie do šiestich zhlukov. Analýzy iných počtov zhlukov sú dostupné k náhľadu na online github repozitári tejto práce[2].

Na Obr. 5 – 4 vidíme ako vyzerali jednotlivé zhluky. Vizualizované udalosti predstavujú prvých 10 členov zhluku. Vizuálnou kontrolou sme zistili rozdelenie udalostí do jednotlivých zhlukov hlavne podľa ich intenzity. Zhluky 1, 4 a 5 obsahujú udalosti väčšej intenzity a naopak zhluky 2 a 3 udalosti nižšej intenzity. Zhluk 0 na

[2] `https://github.com/SamuelJas/DP_Samuel`

prvý pohľad vyzerá, že obsahuje veľmi špecifický typ udalostí, ktoré sa v iných zhlukoch nevyskytujú. Po vizuálnom vyhodnotení sme čistotu jednotlivých zhlukov analyzovali pomocou údajov obsiahnutých vo výstupnej tabuľke. Pozreli sme sa na početnosti udalostí v zhlukoch a ich rozloženie podľa typov.



Obrázok 5 – 4 k-Means – Zhlukovanie do 6 zhlukov

Zhluk	Početnosť udalostí
Zhluk 0	1627
Zhluk 1	1666
Zhluk 2	3487
Zhluk 3	3417
Zhluk 4	1419
Zhluk 5	2031

Tabuľka 5 – 1 Početnosť udalostí v zhlukoch

Z tabuľky 5 – 1 vidíme, že najpočetnejšie sú zhluky 2 a 3. Naopak zhluky 0, 1 a 4

sú menej početné. Na prvý pohľad sa tieto zhluky javia, že obsahujú špecifickejšie udalosti. To sme overili analyzovaním rozloženia sferikov a tweekov v jednotlivých zhlukoch. Analýza je znázornená tabuľkou 5 – 2. Z tejto tabuľky vidíme rovnomerné rozloženie sferikov vo väčšine zhlukov, okrem zhluku 4, kde ich je výrazne menej. Naopak v tomto zhluku 4 je koncentrovaná väčšina tweekov. V zhlukoch 2 a 5 sa naopak nachádza minimum tweekov. Z pohľadu tejto analýzy je však najzaujímavejší zhluk 0, ktorý je tvorený výhradne sferikmi. Môžeme teda povedať, že zhluk 0 je čistým zhlukom sferikov a taktiež zhluky 2 a 5 sú takmer čistými zhlukmi sferikov.

Typ udalosti	Zhluk	Početnosť
Sferik	Zhluk 0	1627
	Zhluk 1	1526
	Zhluk 2	3449
	Zhluk 3	3298
	Zhluk 4	488
	Zhluk 5	2011
Tweek	Zhluk 0	0
	Zhluk 1	140
	Zhluk 2	38
	Zhluk 3	119
	Zhluk 4	931
	Zhluk 5	20

Tabuľka 5 – 2 Rozloženie sferikov a tweekov v zhlukoch

Ďalšou metrikou, ktorú sme sledovali bolo rozloženie udalostí z hľadiska parametra ich emisie f_min<2kHz. Analýza reprezentovaná tabuľkou 5 – 3 nám ešte viac špecifikuje udalosti v jednotlivých zhlukoch. Zhluk 0, o ktorom už vieme, že je čistý zhluk sferikov, táto analýza rozširuje o poznatok, že takmer všetky tieto sferiky majú emisiu nad $2kHz$. Spolu s vizuálnou kontrolou teda vieme povedať, že tento zhluk

obsahuje veľmi špecifický typ udalostí, ktoré sa v ostatných zhlukoch nenachádzajú. Zhluk 4 je presne opačným prípadom. Vieme, že tam prevládajú tweeky a rovnako v tomto zhluku prevláda aj emisia udalostí pod $2kHz$. Zhluky 1, 2, 3 a 5 obsahujú dostatočné zastúpenie oboch tried takže v týchto zhlukoch nie je informácia o emisii $2kHz$ smerodajná.

Parameter f_min<2kHz	Zhluk	Početnosť
Nad $2kHz$	Zhluk 0	1561
	Zhluk 1	456
	Zhluk 2	2650
	Zhluk 3	2174
	Zhluk 4	183
	Zhluk 5	792
Pod $2kHz$	Zhluk 0	66
	Zhluk 1	1210
	Zhluk 2	837
	Zhluk 3	1243
	Zhluk 4	1236
	Zhluk 5	1239

Tabuľka 5 – 3 Rozloženie udalostí podľa emisie $2kHz$ v zhlukoch

Poslednou sledovanou hodnotou bolo rozdelenie udalostí podľa počtu disperzií do jednotlivých zhlukov. Z analýzy zobrazenej na tabuľke 5 – 4 však nie je vidieť žiadne jednoznačné rozdelenie podľa počtu disperzií. Ak je zhluk majoritne tvorený tweekmi, bude obsahovať tweeky s rôznym počtom disperzií. Neukazuje sa žiadny zhluk, ktorý by obsahoval napríklad len tweeky s jednou disperziou a žiadny iný počet disperzií. Počet disperzií preto pravdepodobne nebol hlavným zhlukovacím kritériom k-Means zhlukovania.

Počet disperzií	Zhluk	Početnosť	Počet disperzií	Zhluk	Početnosť
0 disperzií	Zhluk 0	1627	1 disperzia	Zhluk 0	0
	Zhluk 1	1526		Zhluk 1	39
	Zhluk 2	3449		Zhluk 2	5
	Zhluk 3	3298		Zhluk 3	26
	Zhluk 4	488		Zhluk 4	95
	Zhluk 5	2011		Zhluk 5	10
2 disperzie	Zhluk 0	0	3 disperzie	Zhluk 0	0
	Zhluk 1	10		Zhluk 1	13
	Zhluk 2	1		Zhluk 2	10
	Zhluk 3	13		Zhluk 3	37
	Zhluk 4	116		Zhluk 4	181
	Zhluk 5	2		Zhluk 5	2
4 a viac disperzií	Zhluk 0	0			
	Zhluk 1	78			
	Zhluk 2	22			
	Zhluk 3	43			
	Zhluk 4	539			
	Zhluk 5	6			

Tabuľka 5 – 4 Rozdelenie udalostí podľa počtu disperzií v zhlukoch

Po sumarizovaní vykonaných analýz môžeme o zhlukoch povedať nasledovné:

- **Zhluk 0** – Obsahuje špecifické udalosti sferkov s emisiou nad $2kHz$.
- **Zhluk 1** – Obsahuje tweeky a sferiky silnej intenzity s prítomným šumom. Šum môže naznačovať zachytenie udalostí počas silnej búrky.
- **Zhluk 2** – Obsahuje sferiky nízkej intenzity.
- **Zhluk 3** – Prevládajú sferiky ale obsahuje aj časť tweekov. Udalosti sú nízkej

intenzity a oproti zhluku 2 obsahujú vizualizované udalosti modrú farbu v pozadí, čo by mohlo naznačovať, že sa jedná o osamotené udalosti.

- **Zhluk 4** – Zhluk, v ktorom sa koncentruje majorita tweekov s emisiou siahajúcou pod $2kHz$.
- **Zhluk 5** – Zhluk sferikov silnej intenzity.

Výsledky metód zhlukovania budeme konzultovať s doménovými expertmi, ktorí budú vedieť jednotlivé zhluky popísať ešte lepšie a taktiež budú vedieť určiť, či zhluky obsahujú nejaké zaujímavé informácie, ktoré by mohli viesť k novým objavom. Rozsiahlejšie skúmanie zhlukov bude predmetom ďalšieho budúceho výskumu. Zaujímavou informáciou by mohlo byť napríklad sledovanie výskytu udalostí zo zhlukov, počas jednotlivých dní a hodín v roku.

5.3 SOM

Metódu samo-organizujúcich sa máp sme realizovali pomocou dvoch Python knižníc. Knižnicu `minisom` sme použili na vytvorenie SOM mapy a knižnicu `hyperopt` na optimalizáciu parametrov `sigma` a `learning_rate`, ktoré kontrolujú učiaci proces. Ako už bolo spomenuté v podkapitole 1.2, SOM mapu v našej práci vizualizujeme odlišným spôsobom. Typickým príkladom vizualizácie SOM mapy je obrázok 3–4, na ktorom vidíme premietnuté označenia jednotlivých tried vo forme rôznych geometrických tvarov. Na takúto vizualizáciu je potrebné mať k dispozícii označenia jednotlivých tried, ktoré síce máme obsiahnuté v našej výstupnej tabuľke ale ako sme sa už presvedčili v metóde zhlukovania k-Means vyhodnotenej v podkapitole 5.2.2, hlavným zhlukovacím kritériom je intenzita udalostí a triedy obsiahnuté vo výstupnej tabuľke sa vyskytujú vo viacerých zhlukoch. Vizualizácia podobná Obr. 3 – 4 by nám teda nepriniesla žiadnu výpovednú hodnotu. Rozhodli sme sa preto vizualizovať mapu takým spôsobom, že každá bunka mapy obsahuje najlepšieho reprezentanta zhluku, ktorý prislúcha danej bunke. Takto vidíme aké udalosti sa

približne nachádzajú v jednotlivých častiach mapy. Oproti metóde k-Means máme navyše aj informáciu o podobnosti jednotlivých zhlukov, pretože podobné zhluky budú na mape bližšie pri sebe a odlišné zhluky budú zase vzdialenejšie.

5.3.1 Modelovanie

Vyskúšali sme namodelovať dve veľkosti SOM máp. Prvá mapa mala veľkosť 5×5 a druhá 3×3. Už pri metóde k-Means sme zistili, že optimálny počet zhlukov je šesť. Mapa veľkosti 5×5 však zhlukuje udalosti až do 25 zhlukov. Pri analýze týchto zhlukov sa ukázalo, že niektoré zhluky obsahujú veľmi málo príkladov. Riešením by bolo podobné zhluky spojiť. Je však otázne, ako dobre by sme vedeli identifikovať podobné zhluky na základe dostupných údajov. Preto sme sa rozhodli pokračovať s menšou mapou veľkosti 3×3, ktorá zhlukuje udalosti do deviatich zhlukov. Vytvorenie tejto mapy detailnejšie popíšeme v tejto podkapitole a v podkapitole 5.3.2 interpretujeme jej výsledky.

SOM mapu sme trénovali v troch rôznych konfiguráciách. Prvé trénovanie prebehlo bez optimalizácie parametrov, vytvorenú mapu sme vizualizovali a skontrolovali ako vyzerajú jednotlivé zhluky. Druhé trénovanie prebehlo s optimalizovaným parametrom `sigma` pomocou knižnice `hyperopt`. Metóda optimalizácie bola založená na hľadaní najlepšej hodnoty parametra `sigma` metódou `grid_search`. Maximálny počet vyhodnotení sme nastavili na 200. Po vyhodnotení metóda vrátila parameter `sigma` s najlepšou hodnotou, aktualizovali sme parametre učenia mapy a opäť mapu vizualizovali. V treťom trénovaní sme optimalizovali oba parametre učiaceho procesu. Tentoraz sme nastavili maximálny počet vyhodnotení na 50. Metóda vrátila hodnotu pre novú `sigmu` a `learning_rate` parametre. Problémom však bolo to, že táto metóda nie je optimalizovaná na takú malú veľkosť SOM mapy, s akou sme pracovali. Výsledná `sigma` bola teda príliš veľká vzhľadom na dimenzie mapy a mapa po vizualizácií nevykazovala dobré výsledky. Rozhodli sme sa teda na učenie konečnej mapy použiť parameter `sigma` vrátený prvou optimalizačnou metódou a parameter `learning_rate` vrátený druhou optimalizačnou metódou. Vizualizovaná

mapa natrénovaná na týchto parametroch vykazovala najlepšie formovanie zhlukov. Trénovanie mapy bolo realizované príkazom,

```
som = train_som(x, y, input_len, sigma, learning_rate, seed)
```

kde boli jednotlivé parametre nastavené nasledovne:

- `x` = 3 (veľkosť x súradnice mapy)
- `y` = 3 (veľkosť y súradnice mapy)
- `input_len` = 18000 (veľkosť vstupu)
- `sigma` = 0.7005065527450522 (hodnota vrátená prvou optimalizačnou metódou)
- `learning_rate` = 2.11865332222016 (hodnota vrátená druhou optimalizačnou metódou)
- `seed` = 42 (hodnota `seed` pre reprodukovateľnosť výsledkov)

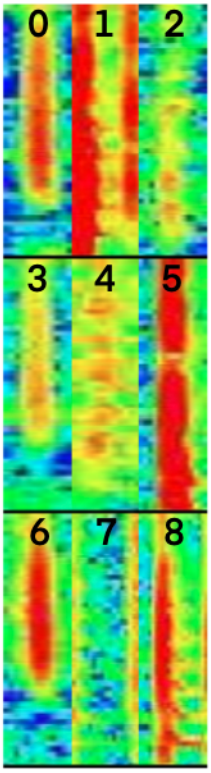
Po vizualizovaní mapy sme z nej extrahovali informáciu o príslušnosti príkladov v jednotlivých zhlukoch. Túto informáciu sme rovnako ako v prípade metódy k-Means spojili s výstupnou tabuľkou pre potreby analýzy a interpretácie zhlukov.

5.3.2 Interpretácia výsledkov

Obr. 5 – 5 predstavuje SOM mapu vytvorenú postupom popísaným v podkapitole 5.3.1. Ako už bolo spomenuté, jednotlivé bunky predstavujú zhluky a vizualizované udalosti v bunkách sú najlepší reprezentanti daných zhlukov. Číselný údaj na obrázku vyjadruje poradie zhluku.

Vizuálnou kontrolou zhlukov vidíme, že na ľavej strane mapy sa opäť koncentrujú špecifické typy udalostí, aké boli nájdené zhlukovaním k-Means. Narozdiel od k-Means metódy sú však rozdelené do troch podzhlukov. Pravá strana mapy obsahuje intenzívnejšie udalosti, na prvý pohľad typu tweek. Zhluky 2 a 4 obsahujú menej

intenzívne udalosti s rozdielom prítomnosti modrej farby v zhluku 2, čo môže opäť značiť zachytenie osamotených udalostí. Po vizuálnej kontrole zhlukov sme podobne ako v prípade k-Means vyhodnotili čistotu zhlukov pomocou dodatočných informácií z výstupnej tabuľky.



Obrázok 5 – 5 Vizualizovaná SOM mapa veľkosti 3×3

Z tabuľky 5 – 5 vieme vyčítať, že jednoznačne najpočetnejším zhlukom je zhluk 4, ktorý obsahuje 6011 udalostí. Naopak zhluk 7 obsahuje veľmi málo udalostí, ale ak sa pozrieme na reprezentanta tohto zhluku vidíme, že je to špecifický druh udalosti, ktorý ani nepredstavuje zachytenú udalosť ale skôr chybu detekcie. Vidíme však, že takýchto udalostí je veľmi malé množstvo. Ďalšie menej početné zhluky sú zhluk 1, 6 a 8.

Ak sa pozrieme na rozloženie tweekov a sferikov v zhlukoch vidíme opäť, že

Zhluk	Početnosť udalostí
Zhluk 0	1117
Zhluk 1	376
Zhluk 2	1065
Zhluk 3	1067
Zhluk 4	6011
Zhluk 5	3434
Zhluk 6	361
Zhluk 7	19
Zhluk 8	197

Tabuľka 5 – 5 Početnosť zhlukov SOM mapy

zhluky 3 a 6 sú čisté zhluky sferikov a zhluk 0 obsahuje len 2 tweeky. Jediným zhlukov kde prevažuje početnosť tweekov nad sferikmi je zhluk 8 a v zhluku 5 sa nachádza majorita z celkového počtu tweekov. Analýzu rozloženia početnosti tweekov a sferikov znázorňuje tabuľka 5 – 6.

Posledná analýza, ktorú sme vykonali bola analýza rozloženia udalostí z hľadiska emisie $2kHz$. Rozloženie je popísané v tabuľke 5 – 7 a vidíme z neho, že zhluky 3 a 6 sú takmer čisté zhluky tvorené udalosťami s emisiou nad $2kHz$. Zhluk 0, ktorý na prvý pohľad vyzerá veľmi podobne ako zhluky 3 a 6 však obsahuje až 249 udalostí s emisiou pod $2kHz$. Ak sa však pozrieme lepšie na vizualizovaného reprezentanta zhluku 0, tak oproti reprezentantom zhlukov 3 a 6 sa emisia udalosti javí ako dlhšia a tým pádom niektoré príklady tohoto zhluku môžu mať emisiu pod $2kHz$. Zhluk 8, ktorý je tvorený majoritne tweekmi obsahuje iba 8 udalostí s emisiou nad $2kHz$. Ďalšie zaujímavé vzory z hľadiska emisie $2kHz$ sa v zostávajúcich zhlukoch nevyskytujú a sú v nich zastúpené oba druhy emisií v dostatočnom počte.

Zhrnutím vykonaných analýz SOM zhlukovania sme zistili, že aj SOM metóda dokázala nájsť špecifické udalosti sferikov s emisiou nad $2kHz$. Navyše však tieto

Typ udalosti	Zhluk	Početnosť udalostí
Sferik	Zhluk 0	1115
	Zhluk 1	301
	Zhluk 2	971
	Zhluk 3	1067
	Zhluk 4	5728
	Zhluk 5	2790
	Zhluk 6	361
	Zhluk 7	18
	Zhluk 8	48
Tweek	Zhluk 0	2
	Zhluk 1	75
	Zhluk 2	94
	Zhluk 3	0
	Zhluk 4	283
	Zhluk 5	644
	Zhluk 6	0
	Zhluk 7	1
	Zhluk 8	149

Tabuľka 5 – 6 Rozloženie sferikov a tweekov v zhlukoch SOM mapy

udalosti rozdelilo do troch podkategórií. Podkategórie boli rozdelené podľa intenzity a dĺžky udalosti. Ďalšiu informáciu, ktorú vieme vyčítať zo SOM mapy je podobnosť zhlukov. Kým na ľavej strane mapy máme udalosti typu sferik s emisiou nad $2kHz$, na pravej strane máme zhluky koncentrujúce tweeky a emisiu prevažujúcu pod $2kHz$.

V porovnaní s metódou k-Means, SOM metóda vnáša nové zaujímavé informácie do zhlukovania o podobnosti jednotlivých zhlukov a taktiež iným prístupom k zhlukovaniu ukazuje nový pohľad na dáta. Vytvorené zhluky sú veľmi podobné zhlukom

vytvoreným metódou k-Means ale špecifické triedy boli v SOM metóde rozdelené na ešte špecifickejšie podtriedy. Mimo špecifických zhlukov 0, 3, 6, 7 a 8 vyzerajú byť zostávajúce zhluky vytvorené na základe intenzity udalostí. Intenzita udalostí je pre nás a aj pre doménových expertov zaujímavá sledovaná metrika, pretože sa podľa nej dá skúmať vzdialenosť búrok, v ktorých boli udalosti zachytené.

Parameter f_min<2kHz	Zhluk	Početnosť
Nad 2kHz	Zhluk 0	868
	Zhluk 1	81
	Zhluk 2	668
	Zhluk 3	1027
	Zhluk 4	3911
	Zhluk 5	903
	Zhluk 6	338
	Zhluk 7	12
	Zhluk 8	8
Pod 2kHz	Zhluk 0	249
	Zhluk 1	295
	Zhluk 2	397
	Zhluk 3	40
	Zhluk 4	2100
	Zhluk 5	2531
	Zhluk 6	23
	Zhluk 7	7
	Zhluk 8	189

Tabuľka 5 – 7 Rozloženie udalostí podľa emisie $2kHz$ v zhlukoch SOM mapy

6 Záver

Na začiatku práce sme si určili viacero cieľov, ktoré sme chceli dosiahnuť. Jedným z nich bolo rozšírenie predchádzajúceho výskumu publikovanom v článku Maslej-Krešňáková et al. (2021) o model založený na báze hlbokých neurónových sietí, ktorý by dokázal detegované udalosti klasifikovať podľa počtu ich disperzií. Vytvorený model dosiahol dobré výsledky a podarilo sa nám ho úspešne spojiť s predchádzajúcim výskumom. Dostali sme tak komplexný súbor metód na automatickú analýzu nameraných dát prístrojom ELMAVAN-G. Tu by sme radi zdôraznili dôležitosť tohoto výskumu, pretože doteraz zachytené udalosti meracím prístrojom neboli nijako automaticky analyzované. V kapitole 4.4 sa nám podarilo vytvorenými metódami analyzovať jeden celý rok nameraných dát a dostali sme prvé reálne informácie o zachytených udalostiach počas dlhšieho časového obdobia. To považujeme za veľký úspech, ktorý výrazne zjednoduší budúce analýzy a dúfame, že povedie k novým objavom. Už nami vytvorené vizualizácie nameraných dát reprezentované Obr. 4 – 9 ukazujú zaujímavé rozloženie jednotlivých druhov udalostí, ktoré určite stoja za ďalšiu analýzu doménovými expertmi. Ak sa ukáže frekvenčný nárast tweekov počas dňa ako pravdivý, mohlo by ísť o úplne nový objav nakoľko vieme, že tweeky sa počas dňa takmer nevyskytujú. V prípade falošnej detekcie by sme zase vedeli poukázať na chybné merania meracieho prístroja, ktoré ostali nepovšimnuté. Ďalším smerovaním výskumu by mohlo byť nasadenie vytvoreného postupu detekcie a klasifikácie udalostí priamo na merací prístroj a následne by sa mohli zachytené udalosti spracovávať a vyhodnocovať v reálnom čase.

Druhým vytýčeným cieľom bolo použitie metód nekontrolovaného učenia na namerané dáta s cieľom objaviť nové alebo nepovšimnuté vzorce a súvislosti v dátach. Motiváciou pre nás bolo to, že metódy nekontrolovaného učenia sa v doméne fyzikálnych dát takmer nepoužívajú. Metódy teda predstavujú potenciál pre nové objavy. Vytvorenými metódami zhlukovania k-Means a SOM sme dokázali v jednotlivých zhlukoch izolovať zaujímavé a špecifické druhy dát. Vytvorené metódy navyše ne-

kladú veľký dôraz na to, či sa jedná o sferiky alebo tweeky, ale udalosti zhlukujú hlavne podľa ich intenzity. Intenzitu dát sme v predchádzajúcom výskume vôbec neskúmali, takže je to pre nás nová a rozširujúca informácia o zachytených udalostiach. V ďalšom výskume by nám mohli vytvorené metódy zhlukovania poslúžiť na jemnejšie delenie kategórií, získaných metódami kontrolovaného učenia.

Autorovi práca priniesla cenné poznatky a skúsenosti pri práci s programovacím jazykov Python. V práci boli realizované metódy kontrolovaného aj nekontrolovaného učenia, pracovalo sa aj s už vytvorenými riešeniami v predchádzajúcom výskume a všetky tieto vytvorené metódy bolo potrebné integrovať, aby spolu dokázali fungovať ako celok. To predstavovalo obtiažnu úlohu, ktorú sa však podarilo splniť. Navyše sa v práci pracovalo s veľkým objemom dát, čo predstavuje výzvu samo o sebe. Všetky tieto nazbierané skúsenosti predstavujú pre autora prínos a bude ich môcť využiť v ďalších vedeckých výskumoch.

Literatúra

Akerlof, C. W., Kehoe, R., McKay, T., Rykoff, E., Smith, D., Casperson, D., McGowan, K., Vestrand, W., Wozniak, P., Wren, J. et al. (2003). The rotse-iii robotic telescope system, *Publications of the Astronomical Society of the Pacific* **115**(803): 132.

Ayyadevara, V. K. (2018). *Convolutional Neural Network*, Apress, Berkeley, CA, pp. 179–215.

Brett, D. R., West, R. G. and Wheatley, P. J. (2004). The automated classification of astronomical light curves using kohonen self-organizing maps, *Monthly Notices of the Royal Astronomical Society* **353**(2): 369–376.

Budden, K. G. (1961). *The wave-guide mode theory of wave propagation*, Logos Press.

Dougherty, G. (2013). *Unsupervised Learning*, Springer New York, New York, NY, pp. 143–155.

El Naqa, I. and Murphy, M. J. (2015). *What is Machine Learning?*, Springer International Publishing, Cham, pp. 3–11.

Fanioudakis, L. and Potamitis, I. (2017). Deep networks tag the location of bird vocalisations on audio spectrograms.

Gao, Y., Liu, W. and Lombardi, F. (2020). Design and implementation of an approximate softmax layer for deep neural networks, *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5.

Ghahramani, Z. (2004). *Unsupervised Learning*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 72–112.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Gupta, N. et al. (2013). Artificial neural network, *Network and Complex Systems* **3**(1): 24–28.

Haq, Z. A. and Jaffery, Z. A. (2021). Impact of activation functions and number of layers on the classification of fruits using cnn, *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 227–231.

Hu, K., Zhang, Z., Niu, X., Zhang, Y., Cao, C., Xiao, F. and Gao, X. (2018). Retinal vessel segmentation of color fundus images using multiscale convolutional neural network with an improved cross-entropy loss function, *Neurocomputing* **309**: 179–191.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization. `https://arxiv.org/abs/1412.6980`, journal=arXiv.org.

Kohonen, T. (1990). The self-organizing map, *Proceedings of the IEEE* **78**(9): 1464–1480.

Konan, O. J. E. Y., Mishra, A. K. and Lotz, S. (2020). Machine learning techniques to detect and characterise whistler radio waves.

Kundrát, A. (2021). *Detekcia bleskov v rádiových spektrogramoch založená na konvolučných neurónových sieťach*, PhD thesis, Technická univerzita v Košiciach.

Manaswi, N. K. (2018). *Convolutional Neural Networks*, Apress, Berkeley, CA, pp. 91–96. `https://doi.org/10.1007/978-1-4842-3516-4_6`.

Maslej-Krešňáková, V., Kundrát, A., Mackovjak, S., Butka, P., Jaščur, S., Kolmašová, I. and Santolík, O. (2021). Automatic detection of atmospherics and tweek atmospherics in radio spectrograms based on a deep learning approach, *Earth and Space Science* **8**(11).

Na, S., Xumin, L. and Yong, G. (2010). Research on k-means clustering algorithm:

An improved k-means clustering algorithm, *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 63–67.

Rumelhart, D. E., Durbin, R., Golden, R. and Chauvin, Y. (1995). Backpropagation: The basic theory, *Backpropagation: Theory, architectures and applications* pp. 1–34.

Santolík, O. and Kolmasova, I. (2017). Unusual electromagnetic signatures of european north atlantic winter thunderstorms, *Scientific Reports* **7**.

Santosh, K., Das, N. and Ghosh, S. (2022). Chapter 2 - deep learning: a review, *in* K. Santosh, N. Das and S. Ghosh (eds), *Deep Learning Models for Medical Imaging*, Primers in Biomedical Imaging Devices and Systems, Academic Press, pp. 29–63. `https://www.sciencedirect.com/science/article/pii/B978012823504100012X`.

Smith, A., Lynn, S. and Lintott, C. (2013). An introduction to the zooniverse, *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Vol. 1.

Srivastava, N. and Srivastava, N. (n.d.). Improving neural networks with dropout.

Vesanto, J. and Alhoniemi, E. (2000). Clustering of the self-organizing map, *IEEE Transactions on Neural Networks* **11**(3): 586–600.

Wang, M., Lu, S., Zhu, D., Lin, J. and Wang, Z. (2018). A high-speed and low-complexity architecture for softmax function in deep learning, *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 223–226.

Wang, Y., Li, Y., Song, Y. and Rong, X. (2020). The influence of the activation function in a convolution neural network model of facial expression recognition, *Applied Sciences* **10**(5). `https://www.mdpi.com/2076-3417/10/5/1897`.

Zhang, Z. (2018). *Artificial Neural Network*, Springer International Publishing, Cham, pp. 1–35.

Zoznam príloh

Príloha A CD médium – záverečná práca v elektronickej podobe, príručky v elektronickej podobe a zdrojový kód.

Príloha B Používateľská príručka

Príloha C Systémová príručka